

Michael Brenner · Nicolas Christin
Benjamin Johnson · Kurt Rohloff (Eds.)

LNCS 8976

Financial Cryptography and Data Security

FC 2015 International Workshops, BITCOIN, WAHC, and Wearable
San Juan, Puerto Rico, January 30, 2015
Revised Selected Papers



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zürich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7410>

Michael Brenner · Nicolas Christin
Benjamin Johnson · Kurt Rohloff (Eds.)

Financial Cryptography and Data Security

FC 2015 International Workshops
BITCOIN, WAHC, and Wearable
San Juan, Puerto Rico, January 30, 2015
Revised Selected Papers



Springer

Editors

Michael Brenner
Leibniz Universität
Hannover
Germany

Nicolas Christin
Carnegie Mellon University
Pittsburgh, PA
USA

Benjamin Johnson
Carnegie Mellon University
Pittsburgh, PA
USA

Kurt Rohloff
New Jersey Institute of Technology
Newark, NJ
USA

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-662-48050-2

ISBN 978-3-662-48051-9 (eBook)

DOI 10.1007/978-3-662-48051-9

Library of Congress Control Number: 2015943047

LNCS Sublibrary: SL4 – Security and Cryptology

Springer Heidelberg New York Dordrecht London

© International Financial Cryptography Association 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

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

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media
(www.springer.com)

BITCOIN 2015: Second Workshop on Bitcoin Research

Building on the success of the First Workshop on Bitcoin Research, which was co-hosted with Financial Cryptography and Data Security 2014, we were delighted to have the opportunity to offer this workshop again this year. If 2013 marked the year in which Bitcoin burst onto the media stage, 2014 was a year of consolidation, in which Bitcoin and distributed payment systems cemented their status as a worthy and practical research topic. Bitcoin's exchange rate, which had been surpassing record high after record high in 2013, settled down to much more modest levels in 2014, driving some of the speculators away.

Meanwhile, researchers identified a number of fascinating challenges brought upon by Bitcoin in domains ranging from legal aspects, to game theory, to distributed system design. Our goal in organizing this second workshop was to continue offering a forum for this exciting research to be heard and discussed. We believe that the program the authors contributed is a good reflection of the diversity of the research Bitcoin can foster.

We gave authors of accepted papers the opportunity to publish the paper *in extenso* in these proceedings or to limit themselves to an abstract if they decided to pursue other avenues for publication. As a result, the proceedings in the volume contain the full, revised versions of nine of the ten accepted papers, and one extended abstract. Authors of Bitcoin-related papers submitted to the main conference were also automatically considered for the workshop. All in all, we considered 15 manuscripts for inclusion in this proceedings volume. These manuscripts were evaluated through a thorough peer-review process, in which each paper was assigned to at least three reviewers.

We thank the authors of all submissions, the members of the Program Committee, and the external reviewers for their efforts; Rainer Böhme and Tyler Moore, the organizers of the first edition, for their sage advice; the presenters and the numerous participants for attending; and the organizers of Financial Cryptography and Data Security 2015 for hosting this workshop. We are especially grateful to the Bitcoin Foundation who acted as Bitcoin-grade sponsor of the main conference and the workshop.

April 2015

Nicolas Christin
Emin Gün Sirer

BITCOIN 2015 Program Committee

Gavin Andresen	Bitcoin Foundation, USA
Elli Androulaki	IBM Zürich, Switzerland
Rainer Böhme	University of Münster, Germany
Joseph Bonneau	Princeton University, USA
Srdjan Capkun	ETH Zürich, Germany
Jeremy Clark	Concordia University, Canada
Stefan Dziembowski	University of Warsaw, Poland
Benjamin Edelman	Harvard University, USA
Ittay Eyal	Cornell University, USA
Christina Garman	Johns Hopkins University, USA
Matt Green	Johns Hopkins University, USA
Joshua Kroll	Princeton University, USA
Sarah Meiklejohn	University College London, UK
Tyler Moore	Southern Methodist University, USA
Andrew Miller	University of Maryland, USA
Roger Wattenhofer	ETH Zürich, Switzerland
Nicholas Weaver	ICSI, USA
Aviv Zohar	Hebrew University of Jerusalem, Israel

WAHC 2015: Third Workshop on Encrypted Computing and Applied Homomorphic Cryptography

The cloud hype and recent disclosures show there is demand for secure and practical computing technologies. The workshop addresses the challenge in safely outsourcing data processing onto remote computing resources by protecting programs and data even during processing. This allows users to outsource computation over confidential information independently from the trustworthiness or the security level of the remote delegate. The workshop serviced these research needs by collecting and bringing together some of the top researchers and practitioners from academia, government, and industry to present, discuss, and share the latest progress in the field relevant to real-world problems with practical approaches and solutions.

The workshop was uniformly attended by academia, government, and industry, with attendees both from prior years with experience in the domain and new attendees learning from the community. Specific encrypted computing technologies focused on homomorphic encryption and secure multiparty computation. The technologies and techniques discussed in this workshop are key to extending the range of applications that can be securely and practically outsourced.

Presentations and discussion at the workshop were of the high quality and deep insight we have come to expect from our community. Topics of conversation included insights and lessons learned from experience implementing encrypted computing schemes, and experience reports on applying these technologies. Special thanks to the invited speakers: Drew Dean from SRI International and Dov Gordon from Applied Communication Sciences, who shared their experiences and involvements from multiple past encrypted computing projects.

The workshop received 16 submissions. All contained unique and interesting results. Each was reviewed by at least three Program Committee members. While all the papers were of high quality, only six papers were accepted to the workshop. We thank the authors for all submissions, the members of the Program Committee for their effort, the workshop participants for attending, and the FC organizers for supporting us.

April 2015

Michael Brenner
Kurt Rohloff

WAHC 2015 Program Committee

Dan Bogdanov	Cybernetica, Estonia
Kevin Butler	University of Florida, USA
David Cousins	BBN, USA
Dario Fiore	IMDEA Software Institute, Madrid, Spain
Shai Halevi	IBM, USA
Vladimir Kolesnikov	Bell Labs, USA
Tancrde Lepoint	CryptoExperts, France
David Naccache	Ecole Normale Superieure, Paris, France
Michael Naehrig	Microsoft, USA
Maire O'Neill	Queen's University Belfast, UK
Pascal Paillier	CryptoExperts, France
Benny Pinkas	Bar-Ilan University, Israel
Christoph Sorge	Universität Saarland, Germany
Osman Ugus	Exceet Secure Solutions, Germany
Yevgeniy Vahlis	University of Toronto, Canada
Marten van Dijk	University of Connecticut, USA
Fre Vercauteren	Katholieke Universiteit Leuven, Belgium
Adrian Waller	Thales, UK

Wearable 2015: First Workshop on Wearable Security and Privacy

Wearable 2015, the First Workshop on Wearable Security and Privacy, was held January 30, 2015, at the InterContinental San Juan Hotel in Isla Verde, Puerto Rico, in association with Financial Cryptography and Data Security 2015. This workshop focused on the unique challenges of security and privacy for wearable devices.

The workshop received eight submissions of which six were accepted. These proceedings contain revised versions of the accepted papers. An invited lecture, entitled “Privacy in the Age of Pervasive Cameras: When Electronic Privacy Gets Physical” was given by Apu Kapadia, Assistant Professor of Computer Science and Informatics at Indiana University.

The Program Committee consisted of 12 members with diverse research interests related to the workshop topic. Each paper was assigned to at least three reviewers. We ensured that each paper received a fair and objective review by experts and also a broader group of Program Committee members. The final decisions on accepted papers were based on reviews and discussion.

We sincerely thank the authors of all submissions. Their efforts gave us an opportunity for a strong and diverse program. We also sincerely thank the efforts of the Program Committee. We are very fortunate that so many brilliant people invested so much time not only in writing reviews, but also in participating actively in follow-up discussions. A list of Program Committee members appears after this note.

April 2015

Benjamin Johnson
John Chuang

Wearable 2015 Program Committee

Alessandro Acquisti	Carnegie Mellon University, USA
Srdjan Capkun	ETH Zürich, Switzerland
John Chuang	University of California, Berkeley (Co-chair), USA
Cory Cornelius	Intel Research, USA
Yves-Alexandre de Montjoye	MIT, USA
Benjamin Johnson	Carnegie Mellon University (Co-chair), USA
Jaeyeon Jung	Microsoft Research, USA
Apu Kapadia	Indiana University, USA
Krishna Ksheerabdhhi	Gemalto, The Netherlands
Ivan Martinovic	University of Oxford, UK
Tara Mathews	Google, USA
Franziska Roesner	University of Washington, USA

Contents

On the Malleability of Bitcoin Transactions	1
<i>Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek</i>	
Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees.	19
<i>Malte Möser and Rainer Böhme</i>	
ZombieCoin: Powering Next-Generation Botnets with Bitcoin	34
<i>Syed Taha Ali, Patrick McCorry, Peter Hyun-Jeen Lee, and Feng Hao</i>	
Cuckoo Cycle: A Memory Bound Graph-Theoretic Proof-of-Work	49
<i>John Tromp</i>	
When Bitcoin Mining Pools Run Dry	63
<i>Aron Laszka, Benjamin Johnson, and Jens Grossklags</i>	
Issues in Designing a Bitcoin-like Community Currency	78
<i>David Vandervort, Dale Gaucas, and Robert St Jacques</i>	
The Bitcoin Market Potential Index	92
<i>Garrick Hileman</i>	
Cryptographic Currencies from a Tech-Policy Perspective: Policy Issues and Technical Directions	94
<i>Emily McReynolds, Adam Lerner, Will Scott, Franziska Roesner, and Tadayoshi Kohno</i>	
Blindcoin: Blinded, Accountable Mixes for Bitcoin	112
<i>Luke Valenta and Brendan Rowan</i>	
Privacy-Enhancing Overlays in Bitcoin	127
<i>Sarah Meiklejohn and Claudio Orlandi</i>	
Search-and-Compute on Encrypted Data	142
<i>Jung Hee Cheon, Miran Kim, and Myungsun Kim</i>	
Accelerating SWHE Based PIRs Using GPUs.	160
<i>Wei Dai, Yarkın Doröz, and Berk Sunar</i>	
Combining Secret Sharing and Garbled Circuits for Efficient Private IEEE 754 Floating-Point Computations	172
<i>Pille Pullonen and Sander Stiim</i>	

Cryptanalysis of a (Somewhat) Additively Homomorphic Encryption Scheme Used in PIR	184
<i>Tancrède Lepoint and Mehdi Tibouchi</i>	
Homomorphic Computation of Edit Distance	194
<i>Jung Hee Cheon, Miran Kim, and Kristin Lauter</i>	
HEtest: A Homomorphic Encryption Testing Framework	213
<i>Mayank Varia, Sophia Yakoubov, and Yang Yang</i>	
Users' Privacy Concerns About Wearables	231
<i>Vivian Genaro Motti and Kelly Caine</i>	
On Vulnerabilities of the Security Association in the IEEE 802.15.6 Standard	245
<i>Mohsen Toorani</i>	
Visual Cryptography and Obfuscation: A Use-Case for Decrypting and Deobfuscating Information Using Augmented Reality	261
<i>Patrik Lantz, Bjorn Johansson, Martin Hell, and Ben Smeets</i>	
Ok Glass, Leave Me Alone: Towards a Systematization of Privacy Enhancing Technologies for Wearable Computing	274
<i>Katharina Krombholz, Adrian Dabrowski, Matthew Smith, and Edgar Weippl</i>	
Design and Analysis of Shoulder Surfing Resistant PIN Based Authentication Mechanisms on Google Glass	281
<i>Dhruv Kumar Yadav, Beatrice Ionascu, Sai Vamsi Krishna Ongole, Aditi Roy, and Nasir Memon</i>	
Glass OTP: Secure and Convenient User Authentication on Google Glass	298
<i>Pan Chan, Tzipora Halevi, and Nasir Memon</i>	
Author Index	309

On the Malleability of Bitcoin Transactions

Marcin Andrychowicz, Stefan Dziembowski,
Daniel Malinowski, and Łukasz Mazurek^(✉)

University of Warsaw, Warsaw, Poland
`{marcin.andrychowicz,stefan.dziembowski,daniel.malinowski,`
`lukasz.mazurek}@crypto.edu.pl`

Abstract. We study the problem of malleability of Bitcoin transactions. Our first two contributions can be summarized as follows:

- (i) we perform practical experiments on Bitcoin that show that it is very easy to maul Bitcoin transactions with high probability, and
 - (ii) we analyze the behavior of the popular Bitcoin wallets in the situation when their transactions are mauled; we conclude that most of them are to some extend not able to handle this situation correctly.
- The contributions in points (i) and (ii) are experimental. We also address a more theoretical problem of protecting the Bitcoin distributed contracts against the “malleability” attacks. It is well-known that malleability can pose serious problems in some of those contracts. It concerns mostly the protocols which use a “refund” transaction to withdraw a financial deposit in case the other party interrupts the protocol. Our third contribution is as follows:
- (iii) we show a general method for dealing with the transaction malleability in Bitcoin contracts. In short: this is achieved by creating a malleability-resilient “refund” transaction which does not require any modification of the Bitcoin protocol.

1 Introduction

Malleability is a term introduced in cryptography by Dolev et al. [15]. Very informally, a cryptographic primitive is *malleable* if its output C can be transformed (“mauled”) to some “related” value C' by someone who does not know the cryptographic secrets that were used to produce C . For example, a symmetric encryption scheme (Enc, Dec) is malleable if the knowledge of a ciphertext $C = \text{Enc}(K, M)$ suffices to compute C' such that $M' = \text{Dec}(K, C')$ is not equal to M , but is related to it (e.g. M' is equal to M with the first bit set to 0). It is easy to see that the standard cryptographic security definitions (like the semantic security of encryption schemes) in general do not imply non-malleability, and hence the non-malleability is usually viewed as an additional (but often highly

This work was supported by the WELCOME/2010-4/2 grant founded within the framework of the EU Innovative Economy (National Cohesion Strategy) Operational Programme. Moreover, Łukasz Mazurek is a recipient of the Google Europe Fellowship in Security, and this research is supported in part by this Google Fellowship.

desirable) feature of the cryptosystems. Since its introduction the concept of non-malleability was studied profoundly, mostly by the theory community, in several different contexts including the encryption and commitment schemes, zero-knowledge [15], multiparty computation protocols [12], hash functions and one-way functions [10], privacy amplification [14], tamper-resilient encoding [16], and many others. Until last year, however, the malleability problem remained largely out of scope of the interests of the security practitioners.

This situation has changed dramatically, when the MtGox Bitcoin exchange suspended its trading in February 2014, announcing that around 850,000 bitcoins belonging to customers were stolen by an attacker exploiting the “malleability of the Bitcoin transactions” [18]. Although there is a good evidence that MtGox used the malleability only as an excuse [13], this announcement definitely raised the awareness of the Bitcoin community of this problem, and in particular, as argued in [13] it massively increased the attempts to exploit this weakness for malicious purposes.

The fact that the Bitcoin transactions are malleable has been known much before the MtGox collapse [21]. Briefly speaking, “malleability” in this case means that, given a transaction T , that transfers x bitcoins from an address A to address B (say), it is possible to construct another transaction T' that is syntactically different from T , but semantically it is identical (i.e. T' also transfers x bitcoins from A to B)¹. This can be done by anybody, and in particular by an adversary who does not know A 's private key. On a high level, the source of the malleability comes from the fact that in the current version of the Bitcoin protocol, each transaction is identified by a hash on its *whole* contents, and hence in some cases such a T' will be considered to be a different transaction than T .

There are actually several ways T' can be produced from T . One can, e.g. exploit the malleability of the signature schemes used in Bitcoin, i.e., the fact that given a signature σ (computed on some message M with a secret key sk) it is easy to compute another valid signature σ' on M (with respect to the same key sk)². Since the standard Bitcoin transactions have a form $T = (\text{message } M, \text{signature } \sigma \text{ on } M)$, thus $T' = (M, \sigma')$ is a valid transaction with the same semantics as T , but syntactically different from T . Another method is based on the fact that Bitcoin permits more complicated transactions than those in the format described above. More precisely, in the so-called “non-standard transactions” the “ σ ” part is in fact a script in the stack based *Bitcoin scripting language*. Therefore, e.g., adding dummy PUSH and POP instructions to σ produces σ' that is operationally equivalent to σ , yet, from the syntactic point of view it is different. See, e.g., [13, 22] for more detailed list of different ways in which the Bitcoin transactions can be mauled.

¹ For a short description of Bitcoin and the non-standard transactions see Sect. 2.

² This is because Bitcoin uses the Elliptic Curve Digital Signature Algorithm (ECDSA) that has the property that for every signature $\sigma = (r, s) \in \{1, \dots, N - 1\}^2$ the value $\sigma' = (r, N - s)$ is also a valid signature on the same message and with respect to the same key as σ .

Recall that in order to place a transaction T on the block chain a user simply broadcasts T over the network. Thus it is easy for an adversary \mathcal{A} to learn T before it is included in the block chain. Hence he can produce a semantically equivalent T' and broadcast T' . If \mathcal{A} is lucky then the miners will include T' into the block chain, instead of T . At the first sight this does not look like a serious problem: since T' is equivalent to T , thus the financial effect of T' will be identical to the effect of T . The reason why malleability may cause problems is that typically in Bitcoin the transactions are identified by their hashes. More precisely (cf., e.g., [8]), an *identifier* (TXID) of every transaction $T = (M, \sigma)$ is defined to be equal to $H(M, \sigma)$, where H is the doubled SHA256 hash function. Hence obviously TXID of T will be different than TXID of T' .

There are essentially three scenarios when this can be a problem. The first one comes from the fact that apparently some software operating on Bitcoin does not take into account the malleability of the transactions. The alleged MtGox bug falls in this category. More concretely, the attack that MtGox claimed to be the victim of looked as follows: (1) a malicious user P deposits x coins on his MtGox account, (2) the client P asks MtGox to transfer his coins back to him, (3) MtGox issues a transaction T transferring x coins to P , (4) the user P launches the malleability attack, obtaining T' that is equivalent to T but has a different TXID (assume that T' gets included into the block chain instead of T), (5) the user complains to MtGox that the transaction was not performed, (6) MtGox checks that there is no transaction with the TXID $H(T)$ and concludes that the user is right, hence MtGox credits the money back to the user's account. Hence effectively P is able to withdraw his coins twice. The whole problem is that, of course, in Step (6) MtGox should have searched not for the transaction with TXID $H(T)$, but for any transaction semantically equivalent to T .

The second scenario is related to the first one in the sense that it should not cause problems if the users are aware that malleability attacks can happen. It is connected to the way in which the procedure of “giving change” is implemented in Bitcoin. Suppose a user A has 3 \AA as an unspent output of the transaction T_0 on the block chain, and he wants to transfer 1 \AA to some other user B. Typically, what A would do in this case is: create a transaction T_1 that has input T_0 and has two outputs: one that can be claimed by B and has value 1 \AA , and the one that can be claimed by himself (i.e. A) and has value 2 \AA . He would then post T_1 on the block chain. If he now creates a further transaction T_2 that claims money from T_1 *without* waiting for T_1 to appear on the block chain, then he risks that T_2 will be invalid, in case someone mauls T_1 .

The third scenario is much more subtle, as it involves the so-called *Bitcoin contracts* which are protocols to form financial agreements between mutually distrusting Bitcoin users. In this paper we assume readers familiarity with this Bitcoin feature. For an introduction to it see, e.g., [3, 4, 19] (in this paper we use the notation from [3, 4]). Recall that contracts are more complicated than normal money transfers in the standard transactions. To accomplish their goal contracts use the non-standard transactions. One of the common techniques used in constructing such contracts is to let the users sign a transaction T_1 before its input T_0 is included in the block chain. Very informally speaking, the problem

is that T_1 is constructed using the TXID $H(T_0)$, which means that an adversary that mauls T_0 into some (equivalent but syntactically different) T'_0 can make the transaction T_1 invalid. There are many examples of such situations. One of them is the “Providing a deposit” contract from [19], which we describe in more detail in Sect. 4, where we also explain this attack in more detail. Note that, unlike in the first two scenarios, this problem cannot be mitigated by patching the software.

1.1 Possible Fixes to the Bitcoin Malleability Problem

There are several ways in which one can try to fix the problems caused by the malleability of Bitcoin transactions. For example one can try to modify Bitcoin in order to eliminate malleability from it. Such proposals have indeed been recently put forward. In particular, Pieter Wuille [22] proposed a number of ad-hoc methods to mitigate this problem, by restricting the syntax of Bitcoin transactions. While this interesting proposal may work in practice, it is heuristic and it comes with no formal argument. In particular, it implicitly assumes that the only way to maul the ECDSA signatures is the one described in Footnote 2, and we are not aware of any formal proof that this is indeed the case.

In our previous paper [3] we proposed another modification of Bitcoin which eliminates the malleability problem. The idea of this modification is to identify the transactions by the hashes of their *simplified* versions (excluding the input scripts). With this modification one can of course still modify the input script of the transaction, but the modified transaction would have the same hash. Unlike [22] this solution does not rely on heuristic properties of the signature schemes. On the other hand, the proposal of [22] may be easier to implement in practice, since it requires milder modifications of the Bitcoin specification.

Another solution proposed recently by Peter Todd [1] is to introduce a new instruction `OP_CHECKLOCKTIMEVERIFY` to the Bitcoin scripting language that allows a transaction output to be made unspendable until some point in the future. It does not concern the problem of malleability directly, but using this opcode would allow to easily create Bitcoin contracts resilient to malleability.

Unfortunately, changing the Bitcoin is in general hard, since it is not controlled by any central authority, and hence the modifications done without proper care can result in a catastrophic fork, i.e. a situation where there is a disagreement among the honest parties about the status of transactions³. Thus, it is not clear if such modifications will be implemented in the close future. It is therefore natural to ask what can be done, assuming that the Bitcoin system remains malleable.

First of all, fortunately, as described above in many cases malleability is not a problem if the software is written correctly, and therefore the most obvious thing to do is the following.

³ An example of such a fork was experienced by the Bitcoin community in March 2013, when it was caused by a bug in a popular mining client software update [11]. Fortunately it was resolved manually, but it is still remembered as one of the moments when Bitcoin was close to collapse.

Direction 1: Educate the Bitcoin software developers about this issue. Convince them that it is a real threat and they should always test their software against such attacks.

The only context in which the malleability cannot be dealt with by better programming are the Bitcoin contracts. Hence a natural research objective is as follows.

Direction 2: Develop a technique that helps to deal with the malleability of Bitcoin transactions in the Bitcoin contracts.

The goal of this paper is to contribute to both of these tasks.

1.2 Our Contribution

The technical contents of this paper is divided into two parts corresponding to the research directions described above. We first focus on “Direction 1” (this is done in Sect. 3). Since most of the software practitioners will probably only care about problems where the threat is real, not theoretic, we executed practical experiments that examine the feasibility of the malleability attacks. It turns out that these attacks are quite easy to perform, even by an attacker that has resources comparable to those of an average user. In fact, our experiments show that it is relatively easy to achieve success rates close to 50 %.

We then analyze the behavior of popular Bitcoin clients when they are exposed to such attacks. Our results indicate that all of them show a certain resilience to such attacks. In particular we did not identify weaknesses that would allow users to steal money (as argued in Sect. 1.3 this is in fact something that one would expect from the beginning). On the other hand, we identified a number of smaller weaknesses in most of these clients. In particular, we observed that in many cases the malleability attack results in making the user unable to issue further transactions, unless he “resets” the client. In most cases such a reset can be performed relatively easy, in one case it required an intervention of a technically-educated user (restoring the backup files), and in two cases there seemed to be no way to perform such action.

This shows that some of the Bitcoin developers seem to still ignore the malleability problem, despite of the fact that over 8 months have passed since the infamous MtGox statement.

The second part (contained in Sect. 5) of this paper concerns the “Direction 2”. We provide a general technique for transforming a Bitcoin contract that is vulnerable to the malleability attacks (but secure otherwise), into a Bitcoin contract that is secure against such attacks. Our method covers all known to us cases of such contracts, in particular, those listed on the “Contracts” page of the Bitcoin Wiki [19], and the lottery protocol of [5]. It can also be applied to [3], what gives the first fair Two-Party Computation Protocol (with financial consequences) for any functionality whose fairness is guaranteed by the Bitcoin deposits, and which, unlike the original protocol of [3] can be used on the current version of Bitcoin⁴.

⁴ The protocol of [3] was secure only under the assumption that the Bitcoin is modified to prevent the malleability attacks.

Related Work. Some of the related work was already described in the previous sections. The idea of using Bitcoin to guarantee fairness in the multiparty protocols and to force the users to respect the outcome of the computation was proposed independently in [3,4] and in [6] (and implicitly in [5]), and was also studied in [7]. The protocols of [4] and [5] work only for specific functionalities (i.e. are not generic), and [5] is vulnerable to the malleability attack. The protocols of [3,6] are generic, but are insecure against the malleability attack. Also the protocol of [7] seems to be insecure against such attacks.

1.3 Ethical Issues

We realize that performing the malleability attacks against the Bitcoin can raise questions about the ethical aspects of our work. We would like to stress that we were only attacking transactions that were issued by ourselves (and we never tried to maul transactions coming from third parties). It is also clear that performing such attacks cannot be a threat to stability of the whole Bitcoin system, since, as reported by [13] Bitcoin remained secure even against attacks on a much higher scale ([13] registered 25,752 individual malleability attacks involving 286,076 bitcoins just on two days of February 2014).

Let us also note that even before we started our work we could safely assume that none of the popular Bitcoin clients is vulnerable to the malleability attacks to the extent that would allow malicious users to steal money, as it is practically certain that any such weakness would be immediately exploited by malicious users. In fact, as argued in [13] such malicious attempts were probably behind the large number of malleability attacks immediately after the MtGox collapse. In other words: the experiments that we performed were almost certainly performed by several hackers before us. We believe that therefore making these results public is in the interest of the whole Bitcoin community.

2 Bitcoin Description

We assume reader’s familiarity with the basic principles of Bitcoin. For general description of Bitcoin, see e.g. [4,17,20]. For the description of non-standard transaction scripts, see [3,4,19]. Let us only briefly recall that the Bitcoin currency system consists of *addresses* and *transactions* between them. An address is simply a public key pk (technically an address is a *hash* of pk). We will frequently denote key pairs using the capital letters (e.g. A). We will also use the following convention: if $A = (sk, pk)$ then $\text{sig}_A(m)$ denotes a signature on a message m computed with sk and $\text{ver}_A(m, \sigma)$ denotes the result (true or false) of the verification of a signature σ on message m with respect to the public key pk .

Each Bitcoin transaction can have multiple inputs and outputs. Inputs of a transaction T_x are listed as triples $(y_1, a_1, \sigma_1), \dots, (y_n, a_n, \sigma_n)$, where each y_i is a hash of some previous transaction T_{y_i} , a_i is an index of the output of T_{y_i} (we say that T_x redeems the a_i -th output of T_{y_i}) and σ_i is called an *input-script*. The outputs of a transaction are presented as a list of pairs $(v_1, \pi_1), \dots, (v_m, \pi_m)$,

where each v_i specifies some amount of coins (called the *value of the i -th output of T_x*) and π_i is an *output-script*. A transaction can also have a time-lock t , meaning that it is valid only if time t is reached. Hence, altogether transaction's most general form is: $T_x = ((y_1, a_1, \sigma_1), \dots, (y_n, a_n, \sigma_n), (v_1, \pi_1), \dots, (v_m, \pi_m), t)$. The *body of T_x* ⁵ is equal to T_x without the input-scripts, i.e.: $((y_1, a_1), \dots, (y_n, a_n), (v_1, \pi_1), \dots, (v_m, \pi_m), t)$, and denoted by $[T_x]$.

One of the most useful properties of Bitcoin is that the users have flexibility in defining the condition on how the transaction T_x can be redeemed. This is achieved by the input- and the output-scripts. One can think of an output-script as a description of a function whose output is Boolean. A transaction T_x defined above is valid if for *every* $i = 1, \dots, n$ we have that $\pi'_i([T_x], \sigma_i)$ ⁶ evaluates to true, where π'_i is the output-script corresponding to the a_i -th output of T_{y_i} . Another conditions that need to be satisfied are that the time t has already passed, $v_1 + \dots + v_m \leq v'_1 + \dots + v'_n$ where each v'_i is the value of the a_i -th output of T_{y_i} and each of these outputs has not been already spent. The scripts are written in the Bitcoin scripting language. Following [4] we will present the transactions as boxes. The redeeming of transactions will be indicated with arrows (cf. e.g. Fig. 3). The transactions where the input script is a signature, and the output script is a verification algorithm are the most common type of transactions and are called *standard transactions*. The address against which the verification is done will be called a *recipient* of this transaction.

We use the security model defined in [4]. In particular, we assume that each party can access the current contents of the block chain, and post messages on it. Let Δ be the maximal possible delay between broadcasting the transaction and including it in the block chain.

3 Experiments

The Implementation of the Malleability Attack. In order to perform malleability attacks we have implemented a special program called **adversary** (using the *bitcoinj* [2] library). This program is connected as a peer to the Bitcoin network and listens for transactions sending bitcoins to a particular address $Addr$ ⁷ owned by us. Whenever such a transaction is received by the program, it mauls the transaction by changing (r, s) into $(r, N - s)$ in the ECDSA signature (cf. footnote 2 on Page 2) and broadcasts the modified version of the transaction.

The effectiveness of the attack is measured by the percent of cases in which the mauled transaction becomes confirmed and the original one invalidated. It depends on the fraction of the network (and hence miners), which receives the mauled transaction before the original one. In order to achieve a high effectiveness we need to push the modified transaction to the whole peer-to-peer network

⁵ In the original Bitcoin documentation this is called “simplified T_x ”.

⁶ Technically in Bitcoin $[T_x]$ is not directly passed as an argument to π'_i . We adopt this convention to make the exposition clearer.

⁷ In our experiments we used addresses 13eb7BFxgHeXfxrdDev1ehrBSGVPG6obu8 and 115g32FHp77hQpuuWpw8j8RYKPvxD1AXyP.

as fast as possible. Therefore, the **adversary** connects to many more peers than a typical Bitcoin client, more precisely it maintains on average 1000 connections⁸. Moreover, it connects directly to some nodes, which are known to be maintained by the mining pools⁹ and sends the mauled transaction to them in the first place.

Effectiveness Analysis. In order to measure the effectiveness of our attack we set up another machine called **victim**, which makes hundreds of transactions sending bitcoins to the address *Addr*. More concretely we measured the effectiveness of mauling transactions under 3 different circumstances:

- (A) The IP address of a victim is not known to the adversary.
- (B) The IP address of a victim is known to the adversary. In this case the adversary can connect directly to the victim, what allows him to discover transactions broadcast by the victim much faster. In our experiments both machines — **adversary** and **victim** were located in the same local network, which in some cases can be possible also in real life (a motivated adversary can connect to the local network used by the victim). Our experiments showed that connecting directly to the victim greatly increases the effectiveness of the attack.
- (C) The victim is aware of the malleability problem and tries to protect against it by broadcasting her transactions on a higher number of connections. In our experiment the victim was connected to 100 peers on the network and her IP address was not known to the adversary.

The results of our experiments are presented on Fig. 1. The effectiveness depends on the nodes to which the **victim** is connected, so after each transaction she drops all her connections and establishes new ones. Moreover, the effectiveness depends on the distribution of mined blocks among miners in the testing period, so experiments were performed over longer periods of time (e.g. 24 hours). In order to exclude the influence of the factors like physical proximity of adversary and victim (in experiments A and C) we performed part of them with **victim** and **adversary** running on machines far away from each other.

Clients Testing. In order to determine the significance of the malleability problem for individual Bitcoin users, we decided to test how the most popular Bitcoin wallets behave when a transaction is mauled. We have tested 14 Bitcoin wallets listed in [9]. In every test we performed several Bitcoin transfers from the wallet to *Addr*. During the tests the **adversary** was trying to maul every transaction addressed to this address.

We observed that (1) most of the clients determine whether the transaction is confirmed by looking for a transaction with a matching hash in the block chain, and (2) the clients are likely to receive from the network the modified transaction

⁸ Typical Bitcoin client maintains about 8 connections.

⁹ More precisely it connects to the nodes maintained by mining pools *GHash.IO* and *Eligius*.

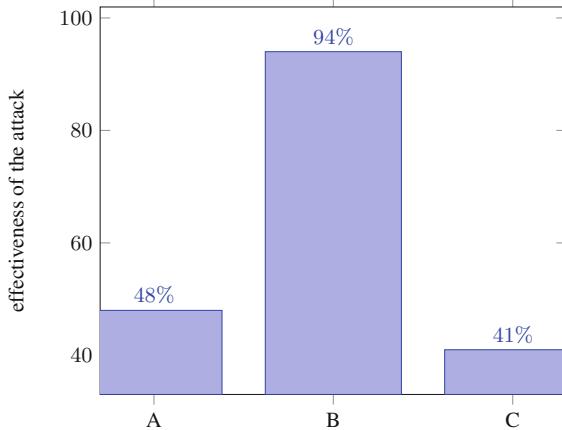


Fig. 1. Effectiveness of the attack under different circumstances.

and therefore list in the transaction history either the original transaction or the modified one or both. This can lead to unpredictable behavior of the wallet of many types, in most cases being hard to precisely describe. In Table 1 we present the results of our tests. In a nutshell, we have observed the following types of behavior of the clients:

- (a) the wallet incorrectly computes the balance,
- (b) the wallet is unable to make an outgoing transaction because it assumes that some transaction will be confirmed in the future (which in fact will never happen),
- (c) the application crashes.

Note that if the client refuses to make an outgoing transaction and it is not possible to remove the troublesome transaction from the history, the user will be unable to spend his bitcoins forever. Moreover, because of the “giving change” procedure, an attack against a single transaction can potentially make all the money in the wallet unspendable (cf. the second scenario on Page 2).

Going more into the details: *Bitcoin Core* and *Xapo* appear to handle the malleability problem correctly. For example *Bitcoin Core* detects the malleability attack and marks it as “double spending” (indicating it with an exclamation mark). *Green Address* and *Armory* display incorrect balances (however, when it comes to allowing a user to make a transaction, they seem to take into account the real balance). *Blockchain.info*, *Coinkite*, *Coinbase*, *Electrum*, *MultiBit*, *Bitcoin Wallet*, and *KnC Wallet* may get stuck waiting for the confirmation of the transaction, which will never be confirmed and hence prevent the user from creating correctly the next transaction. Fortunately, these clients either give the user an option to “synchronize the list of transactions with block chain” or they do it automatically after some time, which makes the problem disappear. In *Hive* we were able to obtain the effect of “resetting” the transaction list only by a manual action (which may be non-trivial for a non-technically educated user).

Table 1. The results of testing 14 Bitcoin wallets listed on [9] against malleability attack. The “X” sign means that the client (a) incorrectly computes the balance, (b) refuses to make an outgoing transaction despite the available funds or (c) crashes. All the tests took place in October 2014 and hence may not correspond to the current software version. Value *never* in the last column means that we could not figure out a way to solve the problem and it did not disappear on its own after a few days.

Wallet name	Type	(a)	(b)	(c)	When the problem disappears
Bitcoin core	Desktop				-
Xapo	Web				-
Armory	Desktop	X			never
Green address	Destop	X			never
Blockchain.info	Web	X	X		after six blocks without confirmation
Coinkite	Web	X	X		after several blocks without confirmation
Coinbase	Web	X	X		after several hours
Electrum	Desktop	X	X		after application reset
MultiBit	Desktop	X	X		after “Reset block chain and transactions” procedure
Bitcoin wallet	Mobile	X	X		after “Reset block chain” procedure
KnC wallet	Mobile	X	X	X	after “Wallet reset” procedure
Hive	Desktop	X	X	X	after restoring the wallet from backup
BitGo	Web	X	X		never
Mycelium	Mobile	X	X		never

The problem is much more severe in case of *BitGo* and *Mycelium*, which also display the troublesome transaction, but there appears to be no way to reset the list. We note that, since BitGo is a web-client, the wrong transaction can be removed by the administrator.¹⁰

4 Malleability in Bitcoin Contracts

As shown in the earlier sections malleability of Bitcoin transactions can pose a problem to users if Bitcoin clients or services they are using have bugs in their implementation. But when these bugs are fixed then there should be no problems or dangers in typical usage of Bitcoin. Unfortunately malleability is a bigger problem for the security of the Bitcoin contracts. In this section we will describe a (known) malleability attack on a protocol for a deposit. Later we will also identify other protocols that are vulnerable to the malleability attack (Fig. 2).

4.1 The Deposit Protocol

The Deposit protocol [19] is executed between parties A and B. To remain consistent with the rest of this paper we describe it here using the notation from [3, 4] (cf. Sect. 2). The idea of this protocol is to allow A to create a financial

¹⁰ In fact, the BitGo administrators reacted to our experiments by contacting us directly.

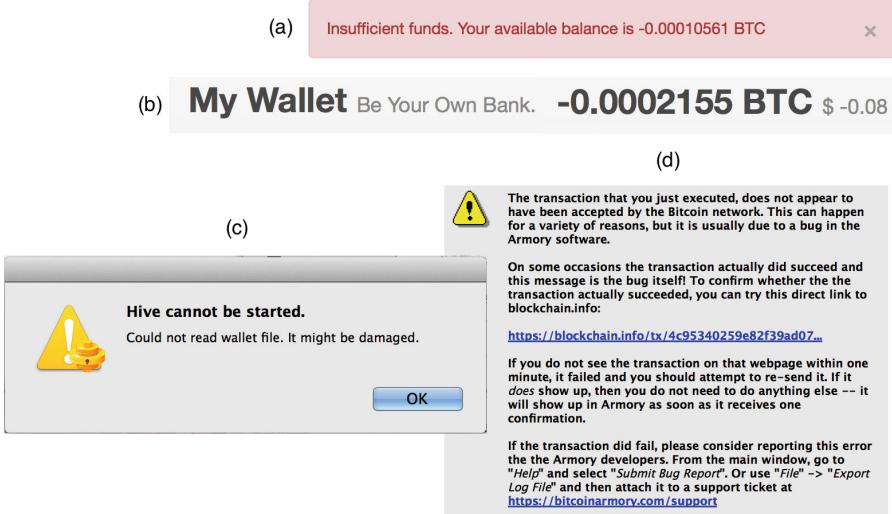


Fig. 2. Different behavior of clients during malleability attacks: (a) BitGo, (b) Blockchain.info., (c) Hive, and (d) Armory.

deposit worth d ₿ for some period of time. A has to be sure that after time t she will get her money back and B has to be sure that A will not be able to claim her money earlier. One of the possible applications of this protocol is the scenario when B is a server and A is a user that wants to open an account on the server B. In this case B wants to be sure that A is a human and not a bot that creates the accounts automatically. To assure that, B forces A to create a small deposit for some time. For an honest user this should not be a big problem, because the deposit is small and she will get this money back. On the other hand it makes it expensive to create many fake accounts (for some malicious purposes), because the cumulative value of the deposits would grow huge.

We will now describe the deposit protocol in an informal way. The main idea of this protocol is fairly simple: A “deposits” her money using a transaction *Deposit* that can be spent only using the signatures of both A and B. To be sure that this money will go back to her she creates a transaction *Fuse* that spends *Deposit*. This transaction needs to be signed by B, and hence A asks B to sign it, and only after A receives this signature she posts *Deposit* on the block chain. In order to prevent A from claiming her money too early *Fuse* contains a timelock t . In more detail the protocol looks as follows:

1. At the beginning A and B exchange their public keys used for signing Bitcoin transactions and they agree on the deposit size d and time t at which the deposit will be freed.
2. Then A creates a transaction *Deposit* of value d ₿, but she does not broadcast it. This transaction is constructed in such a way that to spend it someone has to include both signatures of A and B.

3. Afterwards A creates the transaction *Fuse* that has a time lock t , spends the transaction *Deposit* and sends the money back to her. This transaction is also not yet broadcast.
4. Then A sends the transaction *Fuse* to B, he signs it and sends back to A.
5. Only then A sends the transaction *Deposit* to the block chain.
6. After time t she sends the transaction *Fuse* to get the deposit back.

The graph of transactions in the Simple deposit protocol is presented on Fig. 3. The security properties that one would expect from this protocol are as follows:

- (a) A is not able to get her deposit back before the time t (assuming that B follows the protocol).
- (b) A will not lose her deposit, i.e. she will get $d \text{ } \ddot{\text{B}}$ back before the time $(t + \Delta)$ (where Δ denotes the maximum latency between broadcasting transaction and its confirmation).

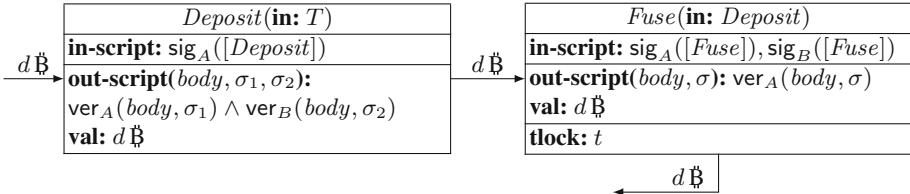


Fig. 3. The Deposit protocol (vulnerable to malleability) from [19].

It is easy to see that (a) holds, since there is no way *Deposit* can be spent before time t (as it requires B's signature to be spent). One would be tempted to say that also (b) holds, since A can always claim her money back by posting *Fuse* on the block chain. Unfortunately, it turns out that this argument strongly relies on the fact that *Deposit* was posted on the block chain *exactly* in the same version as the one that was used to create the *Fuse* transaction. Hence, if the adversary mauls *Deposit* and posts some *Deposit'* (syntactically different, but semantically equivalent to *Deposit*) then the *Fuse* transaction will not be able to spend it (as it expects its input to have TXID equal to $H(\text{Deposit})$, not $H(\text{Deposit}')$).

4.2 Other Protocols Vulnerable to the Malleability Attack

In this section we will list other known Bitcoin contracts that are vulnerable to the malleability attack. The problem with all of them is that they are creating a transaction spending another transaction before the latter is included in the block chain. Each of this protocols can be made resistant to malleability using our technique described in the next section.

- “Example 5: Trading across chains” from [19]¹¹
- “Example 7: Rapidly-adjusted (micro)payments to a pre-determined party” from [19]¹²
- Back and Bentov’s lottery protocol [5]¹³
- *Simultaneous Bitcoin-based timed commitment scheme* protocol from [3]¹⁴

5 Our Technique

In this section we will show how to fix the *Deposit* protocol to make it resistant to malleability. This technique can be used also to other protocols, e.g. those listed in Sect. 4.2. Recall that the reason why the protocols from Sects. 4.1 and 4.2 were vulnerable to the malleability attacks was that one party, say A, had to obtain a signature of the other party (B) on a transaction T_1 , whose input was a transaction T_0 , and this had to happen *before* T_0 was posted on the block chain (in case of the *Deposit* protocol T_0 and T_1 were the *Deposit* and the *Fuse* transactions, resp.). Our main idea is based on the observation that, using the properties of the Bitcoin scripting language, we can modify this step by making T_0 spendable not by using the B’s signature, but by providing a preimage s of some value h under a hash function H (where H can be, e.g., the SHA256 hash function available in the Bitcoin scripting language)¹⁵. In other words, the T_0 ’s spending condition

$$\mathbf{out\text{-}script}(body, \dots, \sigma): \dots \wedge \mathbf{ver}_B(body, \sigma)$$

(cf. the *Deposit* protocol in Fig. 3) would be replaced by

$$\mathbf{out\text{-}script}(body, \dots, x): \dots \wedge H(x) = h,$$

¹¹ The malleability problem occurs in Step 3 when Party A generates Tx2 (the contract) which spends Tx1, and then asks B to sign it and send it back. This happens before Tx1 is included in the block chain and hence if later the attacker succeeds in posting a mangled version Tx1’ of Tx1 on the block chain, then the transaction Tx2 becomes invalid.

¹² The malleability problem is visible in Step 3, where the refund transaction T2 is created. This transaction depends on the transaction T1 that is not included in the block chain at the time when both parties sign it (in Steps 3 and 4).

¹³ The problem occurs in Steps 4 and 7, where the “refund_bet” and “refund_reveal” transactions are signed before their input transactions “bet” and “reveal” are broadcast.

¹⁴ The problem is visible in Step 2 of the *Commit* phase of this protocol (the *Fuse*^A and *Fuse*^B transactions are created before their input transaction *Commit* appears on the block chain).

¹⁵ Such transactions are called *hash locked* in the Bitcoin literature. Notice that having an output script, which requires only preimages and no signatures is not secure, because anyone who notices in the network a transaction trying to redeem such output script learns the preimages and can try to redeem this output script on his own. In our case the output script of the transaction T_0 requires also a signature of A, but we omit it (...) to simplify the exposition.

where $h = H(s)$ is communicated by B to A, and s is chosen by B at random. This would allow A to spend T_0 no matter how it is mauled by the adversary, provided that A learns s . At the first sight this solution makes little sense, since there is no way in which B can be forced to send s to A (obviously in every protocol considered above s would need to be sent to A some time *after* T_0 appears on the block chain, as otherwise a malicious A could spend T_0 immediately). Fortunately, it turns out that this problem can be fixed by adding one more element to the protocol. Namely, we can use the *Bitcoin-based timed commitment scheme* from [4] which is a protocol that does exactly what we need here: it allows one party, called the *Committer* (in our case: B) to *commit* to a string s by sending $h = H(s)$ to the *Recipient* (here: A). Later, B can *open* the commitment by sending s to A (before this happens s is secret to A). The special property of this commitment scheme is that the users can specify a time t until which B has to open the commitment. If he does not do it by this time, then he is forced to pay a fine (in bitcoins) to A. As shown in [4], the Bitcoin-based timed commitment scheme is secure even against the malleability attacks. For completeness we present this protocol in more detail in the next section.

5.1 Bitcoin-Based Timed Commitment Scheme

The Bitcoin-based timed commitment scheme protocol (CS) is being executed between the Committer B and the Recipient A. During the commitment phase the Committer commits himself to some string s by revealing its hash $h = H(s)$. Moreover the parties agree on a moment of time t until which the Committer should open the commitment, i.e. reveal the secret value s . The protocol is constructed in such a way that if the Committer does not open the commitment until time t , then the agreed amount of $d\mathbb{B}$ is transferred from the Committer to the Recipient. More precisely, at the beginning of the protocol the Committer makes a deposit of $d\mathbb{B}$, which is returned to him if he opens the commitment before time t or taken by the Recipient otherwise.

The graph of transactions and the full description of the CS protocol is presented on Fig. 4. The main idea is that if the Committer is honest then only the transactions *Commit* and *Open* will be used (to commit to s and to open s respectively). If, however, the Committer refuses to open his commitment, then the Recipient will post the *Fuse* transaction on the block chain and claim B's deposit. Observe that *Fuse* is time-locked and therefore a malicious recipient cannot claim the money before the time t (and after time t he can do it only if B did not open the commitment). The reader may refer to [4] for more details. Note that even if the transaction *Commit* is maliciously changed before being included in the block chain, the protocol still succeeds because the transaction *Fuse* is created after *Commit* is included in the block chain, so it always contains the correct hash of *Commit*. Therefore, the CS protocol is resistant to the transaction malleability. The properties of the CS protocol are as follows:

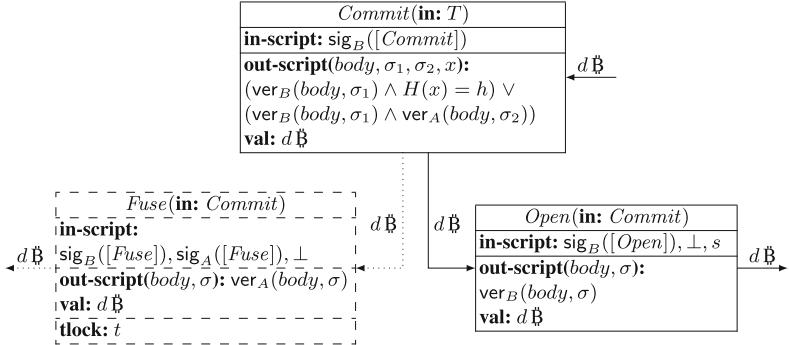


Fig. 4. The CS protocol from [4]. The scripts' arguments, which are omitted are denoted by \perp .

- (a) The Recipient has no information about the secret s before the Committer broadcasts the transaction Fuse (this property is called *hiding*).
- (b) The Committer cannot open his commitment in a different way than revealing his secret s (this property is called *binding*).
- (c) The honest Committer will never lose his deposit, i.e. he will receive it back not later than at the time t .
- (d) If the Committer does not reveal his secret s before the time $(t + \Delta)$ then the Recipient will receive $d \text{฿}$ of compensation.

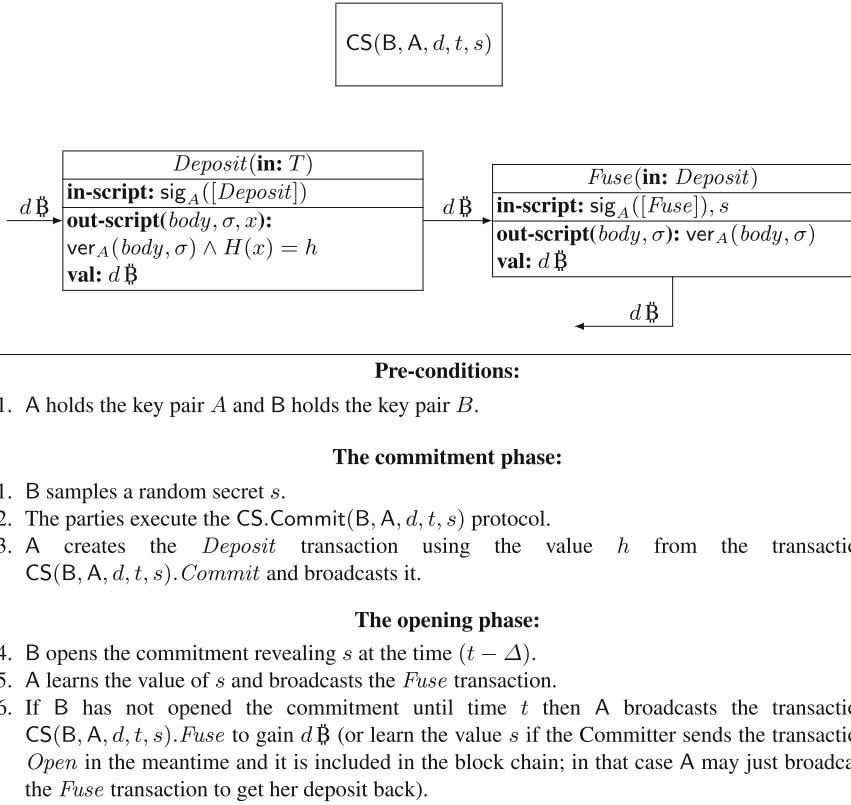


Fig. 5. The solution of the deposit problem resistant to malleability. $\text{CS}(B, A, d, t, r)$ denotes the transactions in the appropriate execution of the CS protocol.

5.2 The Details of Our Method

We now present in more detail our solution of the malleability problem in Bitcoin contracts that was already sketched at the beginning of Sect. 5. It can be used in all of the Bitcoin contracts that are vulnerable to the malleability attacks that we are aware of. In this paper we show how to apply it to the *Deposit* protocol (described in Sect. 4.1).

The main idea of our solution is to use the CS protocol instead of standard *Fuse* transaction. More precisely at the beginning of the protocol B samples a random secret s and commits himself to it using the *Commit* phase of the CS protocol. Now A knows that B will have to reveal his secret (i.e. the value s s.t. $H(s) = h$) before the time t . So A can create a *Deposit* transaction in such a way that to spend it she has to provide her signature and the value s . That means that after the time t either B will reveal the value s and A will be able to spend the transaction *Deposit* or A will get the deposit of B from the CS protocol. Such a modified protocol is denoted *NewDeposit*. Its graph of transactions and its full description is presented on Fig. 5.

The properties of the NewDeposit protocol are as follows (all of them hold even against the malleability attacks):

- (a) A is not able to get her deposit back before the time $(t - \Delta)$.
- (b) The honest A will not lose her deposit, i.e. she will get $d \mathbb{B}$ back before the time $(t + 2\Delta)$.
- (c) Additionally the honest B will not lose his deposit, i.e. he will get it back before the time t .

The proof of the above properties is straightforward and it is omitted because of the lack of space. We note that this protocol may not be well-suited for the practical applications, as it requires the server to make a deposit. Nevertheless, it is a very good illustration of our technique, that is generic and has several other applications.

References

1. bips/bip-0065.mediawiki. <http://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki>. Accessed on 10 December 2014
2. bitcoinj library homepage. <http://bitcoinj.github.io>. Accessed on 20 October 2014
3. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Fair two-party computations via bitcoin deposits. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) Financial Cryptography and Data Security. Lecture Notes in Computer Science, pp. 105–121. Springer, Berlin Heidelberg (2014)
4. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multi-party computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy (SP), May (2014)
5. Back, A., Bentov, I.: Note on fair coin toss via bitcoin (2013). <http://www.cs.technion.ac.il/~7Eiddo/coinossBitcoin.pdf>
6. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014)
7. Bentov, I., Kumaresan, R.: How to use Bitcoin to design fair protocols. Cryptology ePrint Archive, Report 2014/129 (2014). <http://eprint.iacr.org/2014/129>. Accepted to ACM CCS 2014
8. Bitcoin.org. Developer reference. <http://bitcoin.org/en/developer-reference>. Accessed on 20 October 2014
9. Bitcoin.org. List of bitcoin wallets. <http://bitcoin.org/en/choose-your-wallet>. Accessed on 20 October 2014
10. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 524–541. Springer, Heidelberg (2009)
11. Vitalik, B.: Bitcoin network shaken by blockchain fork, March 2013. Bitcoin Magazine. <http://bitcoimmagazine.com/3668/bitcoin-network-shaken-by-blockchain-fork>
12. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th Annual ACM Symposium on Theory of Computing, pp. 494–503. ACM Press (2002)

13. Decker, C., Wattenhofer, R.: Bitcoin transaction malleability and mtgox. In: Kutylowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 313–326. Springer, Heidelberg (2014)
14. Dodis, Y., Wichs, D.: Non-malleable extractors and symmetric key cryptography from weak secrets. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing, pp. 601–610. ACM Press (2009)
15. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. **30**(2), 391–437 (2000)
16. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 239–257. Springer, Heidelberg (2013)
17. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Consulted **1**, 28 (2008)
18. Weisenthal, J.: Bitcoin just completely crashed as major exchange says withdrawals remain halted, Business Insider (2014). <http://www.businessinsider.com/mtgox-statement-on-withdrawals-2014-2>
19. Bitcoin Wiki. Contracts. <http://en.bitcoin.it/wiki/Contracts>. Accessed on 20 October 2014
20. Bitcoin Wiki. Main page. <http://en.bitcoin.it/>. Accessed on 20 October 2014
21. Bitcoin Wiki. Transaction malleability. http://en.bitcoin.it/wiki/Transaction_Malleability. Accessed on 20 October 2014
22. Wuille, P.: Bitcoin improvement proposal: dealing with malleability. <http://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki>. Accessed on 20 October 2014

Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees

Malte Möser¹ and Rainer Böhme^{2(✉)}

¹ Department of Information Systems, University of Münster, Münster, Germany
malte.moeser@uni-muenster.de

² Institute of Computer Science, University of Innsbruck, Innsbruck, Austria
rainer.boehme@uibk.ac.at

Abstract. The Bitcoin protocol supports optional direct payments from transaction partners to miners. These “fees” are supposed to substitute miners’ minting rewards in the long run. Acknowledging their role for the stability of the system, the right level of transaction fees is a hot topic of normative debates. This paper contributes empirical evidence from a historical analysis of agents’ revealed behavior concerning their payment of transaction fees. We identify several regime shifts, which can be largely explained by changes in the default client software or actions of big intermediaries in the ecosystem. Overall, it seems that rules dominate ratio, a state that is sustainable only if fees remain negligible.

1 Introduction

Bitcoin is a protocol claimed to enable a decentralized cryptographic currency [25]. The amount of bitcoin “in circulation”, that is the book value managed in a distributed transaction ledger, is worth about 4–5 billion USD, converted at current market prices [7]. A selling proposition of Bitcoin is that it enables cheap online payments independent of the geographical location of the transaction partners. Therefore, Bitcoin directly competes with established payment systems on the Internet, such as credit cards or PayPal.

Factors influencing the adoption of innovative payment systems are primarily risks and costs [3, 21]. While there is already some work on technical and financial risks of using Bitcoin (e.g., [1, 9, 19, 22–24]), the actual costs of the system are not extensively studied yet. Edelman [13] and Böhme et al. [9] note that, disregarding intangible factors of (in)convenience, Bitcoin may not be as cheap for consumers as it appears. The authors argue that most purchases settled in bitcoin require costly conversions from and to conventional currencies, and consumers forgo kickbacks offered by many credit cards. On top of that, Bitcoin users are encouraged to pay fees to miners, up to 10 cents (of USD) per transaction, irrespective of the amount paid. This is in the same order of magnitude as recently imposed caps on interchange fees for conventional card-based payment systems [12].

Transaction fees are designed to gradually replace the minting revenue as a compensation to miners for contributing to the distributed consensus mechanism that maintains the (probabilistic) consistency of the global system state.

The long-term level of fees is uncertain, yet the question is highly relevant given its connection to the security and sustainability of the system as a whole. Several authors speculate that high fees will render Bitcoin uneconomical for micro payments [14, 20, 29]. Other plausible scenarios include vast variations in fees paid depending on users' time preferences, a low-fee equilibrium with altruistic action to keep the system alive for niche demands [17], or a combination of both with a system of off-blockchain compensation arrangements. Predicting the future regime is hard because it depends not only on properties of the protocol, but also on agent behavior and resulting path dependencies in the Bitcoin ecosystem.

To explore the space of possible future developments, we conduct a longitudinal study of past conventions by analyzing the transaction fees paid with all 55.5 million transactions recorded in the public block chain from the inception of Bitcoin until the end of December 2014. To the best of our knowledge, this first systematic account reveals several regime shifts concerning the payment of transaction fees in Bitcoin's short history. We try to explain these shifts and extract evidence that allows us to test several hypotheses that belong to the conventional wisdom of the Bitcoin community, including:

1. Do higher transaction fees lead to faster confirmation? (yes)
2. Do impatient users offer higher fees? (yes)
3. Do mining pools systematically enforce strictly positive fees? (rather not)

The rest of this paper is structured as follows. Section 2 recalls important properties of Bitcoin with regard to transaction fees. Section 3 documents how we collect and analyze data from the Bitcoin block chain and external sources. Section 4 presents our findings. We discuss limitations and design options for optimal fees in Sect. 5, and conclude in Sect. 6.

2 Background and Research Questions

We refrain from explaining Bitcoin and its terminology in detail and refer the reader to existing high-level [2, 9] or technical [25] descriptions.

Bitcoin's security builds on the block chain, a distributed data structure that allows everyone to look up account balances and to verify unspent transactions.¹ Arguably, the block chain can be seen as a public good, defined by the properties non-excludability and non-rivalry. Exclusion is hard because everyone can anonymously connect to a number of peers and download the block chain. Non-rivalry follows from the block chain being an information good that does not wear out from being shared.

The demand of public goods is characterized by concurrent consumption of all members of a community. This raises issues about the incentives to supply a public good, in particular if the provision incurs costs that cannot be socialized to all members of the community [15]. This is exactly the case in Bitcoin. Miners

¹ More precisely: to verify that all inputs of a transaction one is about to receive reference to so far unspent outputs of past transactions.

unilaterally bear the cost of solving the proof-of-work puzzle,² but all potential transaction partners benefit from the consistency and security of the block chain.

A critical success factor behind Bitcoin's adoption was the reward mechanism that couples, albeit loosely [16], the provision of the public good with newly minted units of currency [8]: 25 BTC per block in 2014. However, this reward mechanism is incompatible with an upper limit of money supply, another stated design goal of Bitcoin. Therefore, the protocol prescribes a transition from minting rewards to transaction fees offered by the sender of a transaction to the miners. By definition, the fee is encoded as difference between the sum of all inputs and the sum of all outputs of a transaction. Miners are free to accept the offer by including the transaction in the block chain, or to ignore it. This creates a market mechanism to find the price of Bitcoin transactions.

In *theory*, perfectly competitive miners will include transactions as long as the fee exceeds the marginal cost of inclusion. Production costs are fixed per block (but may vary between miners depending on access to technology and energy/cooling) and the protocol defines a maximum block size (1 megabyte at the time of writing). As a result, the marginal cost of inclusion is zero if there are fewer unconfirmed transactions than capacity in the block, and it is determined by the opportunity cost of foregone fees from competing transactions as soon as the capacity is reached. Competitive miners make positive expected profits only if transactions compete for space in the block chain. Hence, Houy [17] argues that a maximum block size is necessary for the stability of Bitcoin. However, dominant mining pools or cartels may extract excess profits from reduced competition.

If space in the block chain is scarce and the transaction partners' benefit does not emerge from merely looking up information in the block chain, but depends on the ability to *permanently include* data, then space in the block chain changes its characteristic from a public to something close to a private good. Rivalry comes with the space constraint and excludability with the miners' discretion to exclude unprofitable transactions. However, what remains is that space in the block chain generates substantial externalities: positive ones for parties who benefit from the information and negative externalities for parties who store redundant copies of the block chain in a distributed network.

In *practice*, historical transaction fees in Bitcoin were so small that senders and miners did not care a lot. Many users kept the default value for the transaction fee that is hard-coded in the client software, thereby following a sort of social norm, like for tipping, rather than economic calculus [26]. Likewise, miners followed hard-coded rules [6] to include zero-fee transactions even against their own best interest. Over time, the hard-coded defaults have been changed several times, allegedly to discourage tiny payments (by adding complicated calculation rules) and to offset the rising exchange rate. The latter, in particular, puts consumers' interest over miners', who had to struggle with even steeper increases of the proof-of-work difficulty. A group of programmers went even further and created a fork of the client software that does not offer fees at all [27].

² We suggest that a probabilistic *summation* function, in Hirshleifer's terminology [15], is a reasonable approximation in the short run.

This leads us to the first (open-ended) research question (RQ):

Research Question 1. *How did transaction fees develop and change over time?*

If the client software leaves the users freedom to choose the amount of the transaction fees, then users may follow conventional wisdom about how miners react upon being “tipped” or not.³

Research Question 2. *Do higher fees offered to miners reduce the time until a transaction is first confirmed?*

If RQ 2 is supported with evidence, then it would be rational for users to adjust fees to their time preference.

Research Question 3. *Do impatient users offer higher fees?*

The last research question tests the rationality of the miners, who have no incentive in general to confirm zero-fee transactions. We concentrate on the major mining pools to identify potential differences in their behavior.

Research Question 4. *Do any major mining pools systematically exclude zero-fee transactions?*

In summary, while many are talking about the importance of transaction fees, we are not aware of a comprehensive overview of how fees have changed in the past and why users might decide to deviate from the default. We set out to close this gap with the available data.

3 Data and Method

To study trends of Bitcoin transaction fee conventions over the past couple of years, we combine data from four sources (cf. Table 1). First, we load the block chain by parsing the block files of the Bitcoin Core reference client and extract information on the size of blocks and transactions. To analyze transaction fees as a function of the relation between transactions in the transaction graph, we import all relevant transaction information into an instance of the Neo4j graph database, from which we then extract output amounts, transaction fees, and the duration (based on the blocks’ time stamps) until the first output was reused. We also estimate the net amount of bitcoin transferred, that is total outputs minus estimated change, based on a set of heuristics.

Some analyses require additional data gathered from the website [blockchain.info](#). We use this source to identify the mining pool (if any) that solved a given

³ The default client implements soft rules reflecting part of this wisdom. But uncertainty remains as users cannot anticipate enforcement of these rules. Unlike hard rules (for instance, the requirement to verify signatures), soft rules do not decide the validity of a block. Moreover, miners organized in pools are less likely to heed the defaults than individuals who use the standard client to manage their own transactions.

block and to obtain the time stamps for when a transaction was first seen on the network. This information is not included in the block chain.

Data on the bitcoin exchange rate is taken from [coindesk.com](#), which provides an average bitcoin price in USD. This price index is based on the exchange rates of multiple global exchanges since July 2013, and on the exchange rate of the former exchange Mt. Gox for the time before [11].

Table 1. Data sources and information gathered

Source	Entity	Information
Block files	Block	Height, time stamp, #transactions, size
	Transaction	#inputs, #outputs, size
Graph database	Transaction	Output volume, fee, unused period, net amount, heuristics
Blockchain.info	Block	Relayed by (mining pool)
	Transaction	Time first seen, time included in block
Coindesk.com	Price	USD value

We select the time range from January 2011 to December 2014 for our analysis. Although the Bitcoin block chain exists since 2009, the popularity of the system was low in the first years and interpreting this early data would not be very instructive to understand agent behavior.

In our longitudinal plots, each data point visualizes aggregated data of 1008 blocks, i.e., about one week. The time axis is defined in these epochs of block time with calendar dates added for readability, always using the closest time stamp in the block chain. The constant 1008 was chosen to divide the fixed interval of 2016 blocks of the difficulty control loop that adjusts the proof-of-work requirements. As a result, each pair of consecutive epochs represents blocks mined with the same difficulty. When appropriate, we plot a fitted smoothing spline (with six degrees of freedom) besides the raw data.

To answer RQ 2, we compare the time when a transaction was first seen on the network and the time stamp of the block that includes the transaction. We call the difference transaction latency. Both clocks are not necessarily in sync, but it is reasonable to assume that clock differences are not correlated with our dependent variable. The time when a transaction was first seen on the network has to be extracted by crawling and parsing the [blockchain.info](#) website. To limit the amount of requests, we analyze a representative subset of 9,000 transactions randomly chosen from all eligible transaction between June 2012 and May 2013, a period in which the conventions of fee offers remained relatively stable (see Fig. 3 below). Eligible transactions are defined as transactions that offer a fee of 0, 0.0005, or 0.001 BTC and have a size between 200 and 300 bytes. 60.6 % or, in absolute terms, 9.17 of all 15.1 million transactions in the chosen time range are eligible by these criteria. Limiting our analysis to this homogenous subset removes the need to control for the influence of third variables.

For better comparability, we use the same subset of 9.17 million transactions to answer RQ 3. For each transaction, we compute the holding time, which is the period until one of the outputs was spent again. The computation of this time interval is based on the time stamps of the original block and the block that contains the transaction spending the output. A time interval of zero means, that the output was spent in the same block, i.e., without confirmation. Again, the clocks used for the timestamps of different blocks may not be in sync, but deviations from the true value should not correlate with our variable of interest.

Answering RQ 4 requires information about the mining pool that won each particular block race. We use the information on [blockchain.info](#), parsed from 168,530 HTML pages, as baseline and cross-check against two additional data sources. First, we make use of the fact that some mining pools include a signature in the coinbase transactions of their blocks. This way, we are able to learn the origin of 75,750 blocks mined by the pools 50BTC, AntPool, ASICMiner, BitMinter, BTC Guild, EclipseMC, Eligius, KnCMiner, Polmine, and Slush. This information matches the baseline data from [blockchain.info](#) for 99.98 % of all relevant blocks. A second cross-check against the website [blockorigin.pfoe.be](#), which maps pools to blocks based on the announced blocks on the pools' websites, confirms 99.92 % of the entries. However, this website only provides information for the latest 2016 blocks. All this indicates that our data is pretty reliable when it comes to the attribution of blocks to the major mining pools.

4 Results

4.1 Trends: Descriptive Analysis

We start with an exploratory analysis of transaction fees. The black lines in Fig. 1 show the average sum of transaction fees per block from January 2011 until December 2014. It grew from about 0.1 BTC in early 2012 to 0.25 BTC by mid 2013, with occasional spikes up to 0.5 BTC. In the course of 2014, it fell back to about 0.1 BTC. Overall, miners' revenue from transaction fees is small compared to the minting reward (50 BTC until November 2012, then 25 BTC).

The blue lines visualize the relative transaction fees as percentage of the (estimated) net amount. This value is of interest when looking at the competition between online payment systems. Overall, Bitcoin transaction fees are lower than 0.1 % of the transmitted value, which is significantly below the fees charged by conventional payment systems even if one accounts for the fact that some payments settle in two or more Bitcoin transactions.

The red lines show the average block size (in MB), which grew steadily to about 0.3 MB in December 2014. In the recent past, the default block size limits were increased from 250 KB to 350 KB in September 2013, and from 350 KB to 750 KB in March 2014. Although some blocks get close to the limit, it appears that hard size limits do not (yet) significantly drive the level of transaction fees.

When comparing the total transaction fees per block in USD to the Bitcoin exchange rate against USD, we see substantial co-movement (cf. Fig. 2). This indicates that BTC is the dominant unit of account when deciding about fee

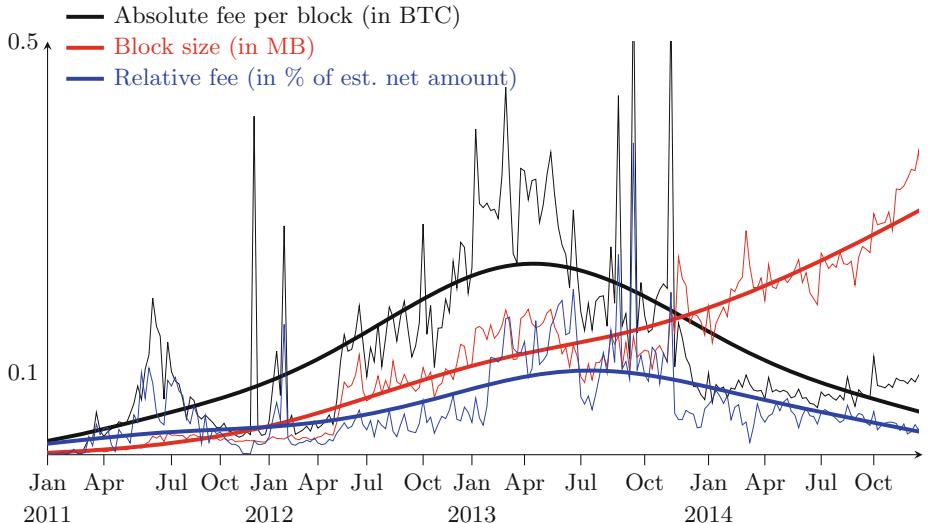


Fig. 1. Transaction fees per block (in BTC) and block size (in MB) (Color figure online)

offers, unlike prices for many goods and services paid with Bitcoin but fixed in a conventional currency. The smoothed curve shows that total fees have stabilized at about 45 USD per block in 2014.

Next, we explore changes in the nominal values of fees. Figure 3 shows trends for the fees paid *per transaction* over time. Each region represents the percentage of transactions with a specific nominal fee. Starting in January 2011, almost no transaction pays a fee. In the following months, a growing share of transactions started to include a fee of 0.01 BTC. The first notable change occurs after June 2011. Transactions with a fee of 0.0005 BTC appear and account for about 20–30 % of all transactions. In the second quarter of 2012, the share of zero-fee transactions drops significantly and 60–70 % of all transactions pay a fee of 0.0005 BTC. In the fourth quarter of 2012, this dominant share registers a sharp decrease, with a fee of 0.001 BTC now accounting for 30–40 % of all transactions. In May 2013, the nominal value of 0.001 BTC makes space for a tenth: 0.0001 BTC. This fee level stays on and gains a share of more than 70 % towards the end of the sample. The second largest nominal fee paid at the time of writing is 0.0002 BTC. This value started to appear in late 2013 and has a share of 15–20 %. It is very evident from Fig. 3 that the conventions on transaction fees are not static, but exhibit distinct trends over time.

In order to reason about these changes, we map important events in the Bitcoin ecosystem to the timeline (cf. Fig. 3). Generally, there seem to be two main reasons for shifts in trends: changes to the Bitcoin reference implementation and actions by large intermediaries in the ecosystem.

The emergence of 0.0005 BTC fees in June 2011 can be mapped to the release of version 0.3.23 of the Bitcoin Core client, which reduced the default transaction

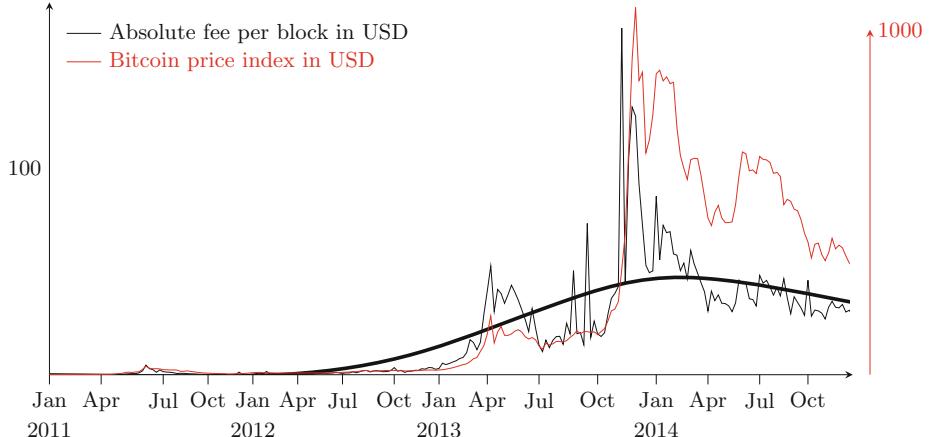


Fig. 2. Transaction fees per block (in USD)

fee from 0.01 BTC to 0.0005 BTC. The rise of transactions with a 0.0005 BTC fee in the second quarter of 2012 is probably due to the launch of the gambling website SatoshiDice. This service works as follows. A user can send a hand-picked amount of bitcoin to an address controlled by SatoshiDice. The service owns several deposit addresses with different associated win and payout ratios. For every incoming transaction, SatoshiDice instantly creates a new transaction to the incoming transaction's source address. This new transaction returns the prize to the user in case he is lucky, or a very small output value to signal a loss.⁴ After its announcement on 24 April 2012, the service quickly gained popularity. It started to flood the block chain with transactions, leading to allegations of being a “DDoS attack against the Bitcoin network” [5].

While we could not find a plausible reason for the drop of the 0.0005 BTC nominal fee in late 2012, we found a possible explanation when looking at the payout transactions of SatoshiDice before and after this shift. Prior to it, SatoshiDice added a transaction fee of 0.0005 BTC to each payment. Then, in the fourth quarter of 2012, it doubled the fee to 0.001 BTC, while everyone else still payed the Bitcoin client's default fee of 0.0005 BTC.

On 29 May 2013, version 0.8.2 of Bitcoin Core, the reference implementation, was released. In this update, the default transaction fee was lowered from 0.0005 BTC per KB to 0.0001 BTC per KB. Hence, the growing share of 0.0001 BTC fees might also visualize the adoption rate of both the new version of the reference client and other clients following this change.

The emergence of 0.0002 BTC fees starting in November 2013 can possibly be attributed to the release of version 1.9 of the Electrum wallet, which set this default fee in order to account for larger transactions due to the use of uncompressed addresses.

⁴ It was one Satoshi initially, then increased to 0.000005460 BTC after the default client required a minimum output value of 0.000005430 BTC to fight transaction spam [10].

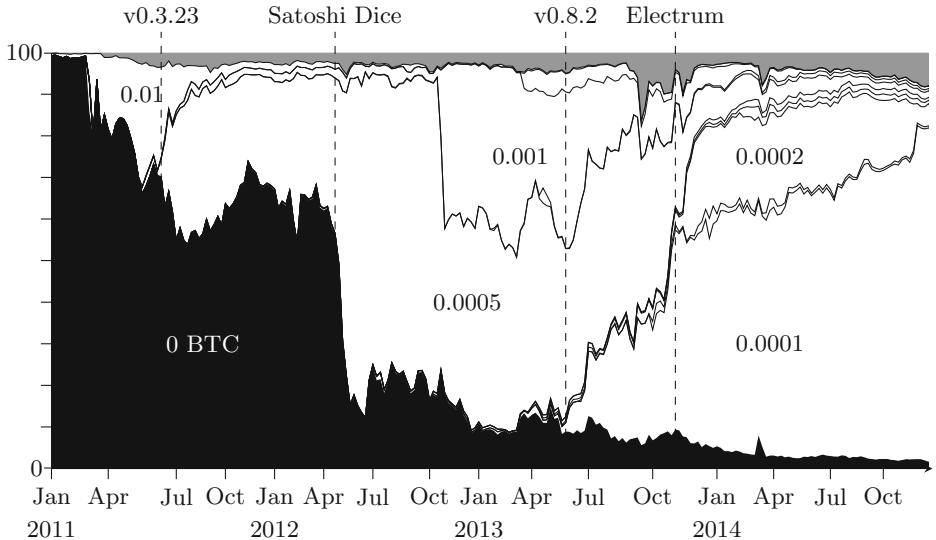


Fig. 3. Distribution of transaction fees

4.2 Tips: Explaining the Decision to Offer a Fee

Even at the time of writing there is a small share of transactions that does not offer a transaction fee to miners. Many of those paying a fee adhere to the default, but some were even willing to pay a higher fee. A plausible rationale is that paying a fee provides incentives for miners to prioritize a transaction, leading to faster confirmation. If this holds true, impatient users would be more willing to pay a fee, i. e., if transaction outputs are to be spent again soon after inclusion.

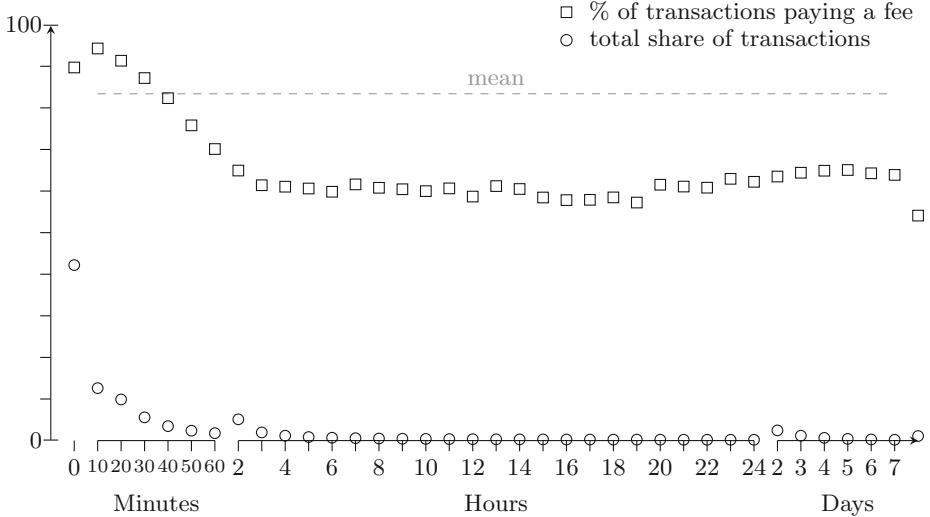
Table 2 shows the quantiles of transaction latencies for different fees. Half of all zero-fee transactions had to wait more than twenty minutes for their first confirmation. In contrast to that, paying a fee of 0.0005 BTC lead to an inclusion into a block in half of the time. While this seems acceptable for less time-critical transactions, the 90 % quantile shows a more extreme difference. Ten percent of all zero-fee transactions took almost 4 hours to confirm, in contrast to 40 min for transactions paying a 0.0005 BTC fee. The difference between paying a fee of 0.0005 or 0.001 BTC is not as pronounced, but the difference in medians is still statistically and economically significant.

Figure 4 reports the amount of transactions that include a fee dependent on the holding time. It is easy to see that the percentage of transactions including a fee is higher for those where outputs are spent shortly after being included in a block. The curve levels off to about 60–70 % for holding times of more than one hour. Another observation is that the amount of transactions whose outputs are reused in the same block amounts to slightly more than 40 %. We suspect that many of these transactions belong to SatoshiDice's zero-confirmation transactions.

Table 2. Transaction latency in seconds by transaction fee

Fee	# Tx	Quantiles of the latency distribution				
		10 %	25 %	50 %	75 %	90 %
0	1503	180	444	1339	4270	13927
0.0005	5735	106	255	600	1244	2440
0.001	1905	90	212	520	1129	2135

Sample period: June 2012 to May 2013. See text for details.

**Fig. 4.** Distribution of holding times and propensity to pay a fee (June 2012–May 2013)

4.3 Tolls: Mining Pools as Gatekeepers

Finally, we analyze pool behavior regarding a possible systematic exclusion of zero-fee transactions. Figure 5 shows the block solution share of each mining pool over time. Shares have shifted between pools quite extensively. In 2013, BTC Guild had a market share of up to 40 %. In 2014 both GHash.IO – which triggered controversial discussions when reaching almost a share of 50 % for a short time (cf. [4]) – and Discuss Fish ousted this pool. Also, the share of “other” pools has risen in 2014. Previous incumbents like Slush or 50BTC have lost popularity. Possible reasons include economic and technical factors, like pool fees, service availability, or robustness against attacks (cf. [18, 28]).

Given the dominance of a few mining pools, we now tackle the question whether some pools systematically enforce fees. Table 3 shows the share of zero-fee transactions as well as the share of blocks without any zero-fee transaction (excluding the always present coinbase transaction) for the ten biggest pools.

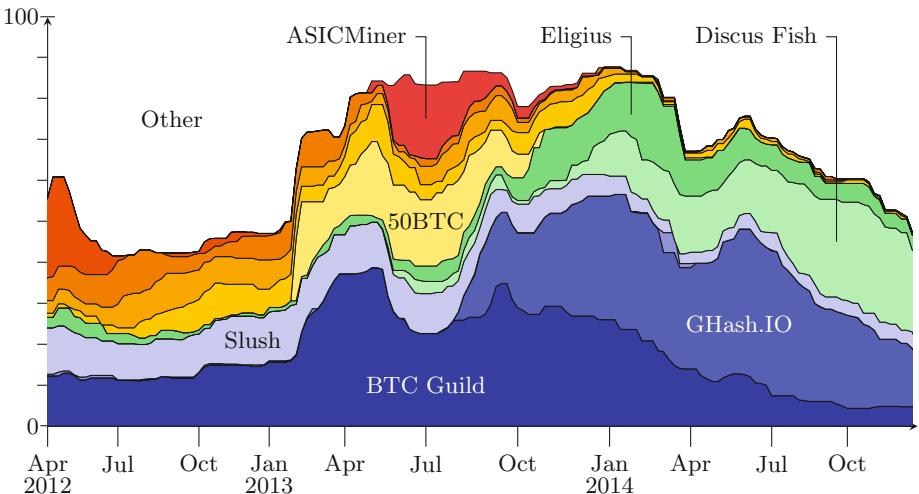


Fig. 5. Block solution share of mining pools

Table 3. Enforcement of transaction fees by mining pools

	Blocks solved (%)	% of zero-fee transactions		% of blocks w/o any zero-fee transaction	
		Apr 2012–	Jan 2014–	Apr 2012–	Jan 2014–
All miners	100.0	7.2	2.7	8.5	17.7
BTC Guild	18.0	6.5	2.2	1.5	4.1
GHash.IO	13.0	4.0	3.4	2.0	2.3
Slush	7.2	5.6	3.4	6.9	2.7
Discus Fish	7.0	0.7	0.3	66.3	72.5
Eligius	5.2	4.1	0.7	26.1	29.2
50BTC	3.9	8.2	11.9	0.4	3.3
BitMinter	3.5	10.4	14.9	3.5	0.8
EclipseMC	3.3	22.3	3.8	2.0	2.6
OzCoin	2.8	8.0	3.3	1.2	7.7
ASICMiner	1.9	8.8	5.9	1.7	0.0

Excluding coinbase transactions. Pool data before Apr 2012 is unreliable.

To account for developments over time, we contrast a longer sample (since April 2012) to the more recent past (since January 2014). The results show that two pools, Discus Fish and Eligius, have a considerably higher share of blocks without any zero fee-transaction: 29.2 % for Eligius and 72.5 % for Discus Fish (since January 2014) – in contrast to an average of 17.7 %. Other than that, there is no clear evidence for enforcement of strictly positive transaction fees.

A reason for the high number of blocks without zero-fee transactions for Discus Fish and Eligius could be that these blocks do not contain any transactions (besides the unavoidable coinbase transaction). Empty blocks appear on the block chain every now and then (one in 117 in 2014). To control for this, we calculate the median number of transactions within these blocks. For Eligius, the median number of transactions amounts to 83, for Discus Fish to 352. Hence, blocks without zero-fee transactions are not completely empty. These two pools seem to take a stricter line at enforcing transaction fees in their blocks than the other big pools.

5 Discussion

We interpret the heterogeneity and instability over time in transaction fees as an indication that the protocol’s built-in market mechanism fails to set a fair price for transactions. This may be tolerable as long as minting rewards dominate miners’ revenue and set the right incentive to defend the system, e.g., by keeping the cost of 51% attacks high at any point in time. Two questions with relevance for the future of Bitcoin remain: 1. What factors influence a fair level of transaction fees? 2. Which mechanism can (approximately) find and enforce this level of fees?

A discussion section of an empirical paper is not the right place for a formal theoretical model. But it is safe to state that a fair price for transactions should internalize the externalities (cf. Sect. 2). Costs to others arise in two forms, born by two different types of agents in the system. First, miners bear the cost of solving the proof-of-work puzzle for the first confirmation. This cost is one-off, fixed per block, and thus depends on the number of transactions seeking confirmation at the same time. Second, relays in the network (that are all clients who store the entire block chain) bear the cost of storing the transaction record. It is current practice to store transactions forever, but in theory records can be pruned after all outputs are spent [25]. Cost of this second kind are incurred over time and depend on the size of the transaction (storage space), the time until all outputs are spent, and the size of the network (number of redundant copies).

It is conceivable that an internalization of the costs of the first kind can be enforced by miners (with some caveats, e.g., [17]). But there remains a free-riding problem regarding the provision of a public good with uninternalized costs of the second kind. Two out of three factors driving the costs of the second kind are not predictable at the time when the transaction is created. Taking averages over many transactions is no solution. It will lead to cherry-picking and other frictions. The time dimension makes it particularly challenging to find a mechanism that internalizes these externalities, as well as the *positive* externalities to longterm investors and potential transaction partners who extract utility from the very existence of the block chain although they rarely make transactions. In a fee-only regime, those with higher transaction demand and time preference subsidize others who can silently sit on their assets. Against this backdrop, it seems that the devaluation of stock, as implemented through monetary inflation

in the minting era, could be a closer approximation of the optimal mechanism than taxing transaction activity.

A limitation of our empirical approach is that off-blockchain payments and other agreements are unobservable. For instance, mining pools could allow exceptions for their own transactions used for reward redistribution or accept other forms of compensation from business partners, such as large intermediaries. (Such compensations are attractive because they can also be hidden from the miners and need not be redistributed.) As a result, what we identify as not rational may indeed be rational under the hidden agenda.⁵ Another limitation is that this initial analysis relies on central moments (mean, median) and subsamples of homogeneous transactions. This hides many particularities in a total of 55.5 million heterogeneous transactions. We suspect that various other factors influence the transaction fees in subsets of transactions too small to isolate in this analysis.

6 Concluding Remarks

A longitudinal analysis of 55.5 million transaction records reveals several regime shifts in agents' behavior related to the payment of transaction fees. This calls for caution against the risk of unobserved heterogeneity in all analyses that do not explicitly consider the time dimension.

Throughout Bitcoin's history, it appears that the level of transaction fees is primarily driven by social norms and conventions formed by key actors in the ecosystem rather than set by the protocol's implied market mechanism, which in principle could match miners' supply with transaction partners' demand. In other words, most agents seem to follow rules instead of economic ratio.

This history, however instructive it may be, is unlikely to offer good predictions for a (distant) future. Fees were generally low in the past, so that agents' ignorance can be explained with information, search, and decision costs. (In simple terms: they do not care.) At least agents will need to revisit their behavior when transaction fees replace minting rewards as the incentive for miners to maintain the system secure. Possibly, the Bitcoin stakeholders may also need to revisit the protocol's incentive system.

Acknowledgements. We thank the anonymous reviewers for their feedback and the hint on the emergence of 0.00002 BTC fees. We also thank blockchain.info for providing us with an API key to bypass their request limit.

References

1. Ali, R., Barrdear, J., Clews, R., Southgate, J.: Innovations in payment technologies and the emergence of digital currencies. In: Digital Currencies: Quarterly Bulletin 2014 Q3. Bank of England (2014)

⁵ We appreciate hints and anecdotes which might lead to testable hypotheses.

2. Ali, R., Barrdear, J., Clews, R., Southgate, J.: The economics of digital currencies. In: Digital Currencies: Quarterly Bulletin 2014 Q3. Bank of England (2014)
3. Anderson, R.: Risk and privacy implications of consumer payment innovation. In: Federal Reserve Bank Payments Conference (2012)
4. Bershadsky, L.: Trust Will Kill Bitcoin (2014). <http://www.bloombergview.com/articles/2014-07-17/trust-will-kill-bitcoin>. Accessed 16 Oct 2014
5. Bitcoin Wiki. SatoshiDice. <https://en.bitcoin.it/wiki/SatoshiDice>. Accessed 17 Sept 2014
6. Bitcoin Wiki. Transaction Fees (2014). https://en.bitcoin.it/w/index.php?title=Transaction_fees&oldid=45501. Accessed 15 Oct 2014
7. Blockchain.info. Bitcoin Market Capitalization (2014). <https://blockchain.info/charts/market-cap>. Accessed 08 Oct 2014
8. Böhme, R.: Internet protocol adoption: learning from Bitcoin. In: IAB Workshop on Internet Technology Adoption and Transition (ITAT), Cambridge, UK (2013)
9. Böhme, R., Christin, N., Edelman, B., Moore, T.: Bitcoin: economics, technology, and governance. *J. Econ. Perspect.* (2014). Available at SSRN: <http://ssrn.com/abstract=2495572>. Forthcoming; Harvard Business School NOM Unit Working Paper No. 15-015
10. Buterin, V.: Bitcoin Developers Adding \$0.007 Minimum Transaction Output Size (2013). <http://bitcoinmagazine.com/4465/bitcoin-developers-adding-0-007-minimum-transaction-output-size/>. Accessed 17 Oct 2014
11. Coindesk. About the Bitcoin Price Index (2014). <http://www.coindesk.com/price/bitcoin-price-index/>. Accessed 23 Oct 2014
12. The Economist. Plastic stochastic (2014). <http://www.economist.com/news/finance-and-economics/21621882-capping-fees-card-transactions-has-not-worked-out-planned-plastic-stochastic>. Accessed 22 Oct 2014
13. Edelman, B.: Consumers Pay More When They Pay with Bitcoin (2014). <http://www.pymnts.com/in-depth/2014/consumers-pay-more-when-they-pay-with-bitcoin/>. Accessed 15 Jan 2015
14. Gavin Andresen comments on Bitcoin 0.8.6 Released, Updates to Block Size Limit, Free Transactions, OSX Bugs (2013). http://www.reddit.com/r/Bitcoin/comments/1sk3df/bitcoin_086_released_updates_to_block_size_limits/cdyrab7. Accessed 13 Oct 2014
15. Hirshleifer, J.: From weakest-link to best-shot: the voluntary provision of public goods. *Public Choice* **41**(3), 371–386 (1983)
16. Houy, N.: The Bitcoin Mining Game. Working Paper GATE 2014–12 (2014)
17. Houy, N.: The Economics of Bitcoin Transaction Fees. Working Paper GATE 2014–07 (2014)
18. Johnson, B., Laszka, A., Grossklags, J., Vasek, M., Moore, T.: Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 72–86. Springer, Heidelberg (2014)
19. Karame, G.O., Androulaki, E., Capkun, S.: Two bitcoins at the price of one? double-spending attacks on fast payments in Bitcoin. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2012)
20. Kaskaloglu, K.: Near zero Bitcoin transaction fees cannot last forever. In: The International Conference on Digital Security and Forensics (DigitalSec 2014), pp. 91–99 (2014)
21. Luther, W.J.: Cryptocurrencies, Network Effects, and Switching Costs. Mercatus Center Working Paper No. 13–17 (2013)

22. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of Bitcoin: characterizing payments among men with no names. In: Proceedings of the ACM Internet Measurement Conference (IMC), pp. 127–140. ACM, New York (2013)
23. Moore, T., Christin, N.: Beware the middleman: empirical analysis of Bitcoin-exchange risk. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 25–33. Springer, Heidelberg (2013)
24. Möser, M., Böhme, R., Breuker, D.: Towards risk scoring of Bitcoin transactions. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 16–32. Springer, Heidelberg (2014)
25. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
26. Sherif, M.: The Psychology of Social Norms. Harper, New York (1936)
27. [UPDATE: 2014-10-05] Bitcoin Core soft-fork “No Forced TX Fee” v0.9.3. <https://bitcointalk.org/index.php?topic=22434.0>. Accessed 23 Oct 2014
28. Vasek, M., Thornton, M., Moore, T.: Empirical Analysis of Denial-of-Service Attacks in the Bitcoin Ecosystem. In: Böhme, Rainer, Brenner, Michael, Moore, Tyler, Smith, Matthew (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 57–71. Springer, Heidelberg (2014)
29. Wong, J.I.: New Study: Low Bitcoin Transaction Fees Unsustainable (2014). <http://www.coindesk.com/new-study-low-bitcoin-transaction-fees-unsustainable/>. Accessed 13 Oct 2014

ZombieCoin: Powering Next-Generation Botnets with Bitcoin

Syed Taha Ali^(✉), Patrick McCorry, Peter Hyun-Jeen Lee, and Feng Hao

Newcastle University, Newcastle upon Tyne, UK

{taha.ali,patrick.mccorry,peter.lee,feng.hao}@ncl.ac.uk

Abstract. Botnets are the preeminent source of online crime and arguably the greatest threat to the Internet infrastructure. In this paper, we present ZombieCoin, a botnet command-and-control (C&C) mechanism that runs on the Bitcoin network. ZombieCoin offers considerable advantages over existing C&C techniques, most notably the fact that Bitcoin is designed to resist the very regulatory processes currently used to combat botnets. We believe this is a desirable avenue botmasters may explore in the near future and our work is intended as a first step towards devising effective countermeasures.

1 Introduction

Almost eight years have passed since Vint Cerf’s dire warning of a botnet “pandemic” [1], and since then the threat has only intensified. Large botnets today typically number millions of infected victims (individually referred to as bots or *zombies*), employed in a wide range of illicit activity including spam and phishing campaigns, spying, information theft and extortion [2]. The FBI recently estimated that 500 million computers are infected annually, incurring global losses of approximately \$110 billion [3]. Botnets have now started conscripting mobile phones [4] and smart devices, such as refrigerators and surveillance cameras to spam and mine cryptocurrencies [5].

The fatal weak point for botnets is the C&C infrastructure, the central nervous system of the botnet. Downstream communication comprises instructions and software updates sent by the botmaster, whereas upstream communication from bots includes *loot*, such as financial data, login credentials, etc. Security researchers usually reverse engineer a bot, infiltrate the C&C network, trace the botmaster and disrupt the botnet. The overwhelming majority of successful takedown operations to date have relied heavily on exploiting or subverting botnet C&C infrastructures [2].

In this paper, we argue that Bitcoin is an ideal C&C dissemination mechanism for botnets. Bitcoin offers botmasters considerable advantages over existing C&C techniques such as chatrooms, HTTP rendezvous points, or P2P networks. First, by piggybacking communications onto the Bitcoin network, the botmaster is spared the costly and hazardous process of maintaining a custom C&C network. Second, Bitcoin provides some degree of anonymity which

may be enhanced using conventional mechanisms like VPNs or Tor. Third, Bitcoin has built-in mechanisms to harmonize global state, eliminating the need for bot-to-bot communication. Capture of one bot therefore does not expose others, and an observer may not easily enumerate the size of the botnet.

Most importantly, C&C communications over the Bitcoin network cannot be shut down simply by confiscating a few servers or poisoning routing tables. Furthermore, disrupting C&C communication would be very hard to do without seriously impacting legitimate Bitcoin users and may *break* Bitcoin. Any form of regulation would be a flagrant violation of the libertarian ideology Bitcoin is built upon [6]. It would also entail significant protocol modification on the majority of Bitcoin clients scattered all over the world.

In this paper, we explore in detail the possibility of running a botnet over Bitcoin. Our goal, of course, is not to empower criminal operations, but to evaluate this threat so that preemptive solutions may be devised. This is in the spirit of existing research efforts exploring emergent threats (such as cryptovirology [7] and the FORWARD initiative [8]). Our specific contributions are:

1. We present ZombieCoin, a mechanism enabling botmasters to communicate with bots by embedding C&C information in Bitcoin transactions.
2. We enumerate various strategies to embed C&C information in transactions and undertake a detailed comparison.
3. We prototype and deploy ZombieCoin and issue C&C commands to bots over the Bitcoin network. Our results show that bots receive and respond in a 5–12 s window.

2 Background

We summarize here the evolutionary path of C&C mechanisms, followed by a brief overview of Bitcoin.

2.1 Botnet C&C Mechanisms

First generation botnets, such as Agobot, SDBot, and SpyBot (observed in 2002–2003) [9], maintain C&C communications over **Internet Relay Chat (IRC) networks**. The botmaster hardcodes IRC server and channel details into the bot executable prior to deployment, and, after infection, bots log on to the specified chatroom for instructions. This method has numerous advantages: the IRC protocol is widely used across the Internet, there are several public servers which botnets can use, and communication is in real-time. However, the network signature of IRC traffic is easily distinguished. More critically, this C&C architecture is centralized. Researchers can reverse-engineer bots, allowing them to eavesdrop in C&C chatrooms, identify the bots and track the botmaster. Researchers also regularly coordinate with law enforcement to legally take down C&C chatrooms,

crippling the entire botnet in just one step. According to insider accounts, two thirds of IRC botnets are shut down in just 24 hours [10].

The second generation of botnets upgraded to **HTTP-based C&C communications**. Examples include Rustock, Zeus and Asprox (observed in 2006–2008). Bots periodically contact a webserver using HTTP messages to receive instructions and offload loot. HTTP is ubiquitous on most networks and bot communications blend in with legitimate user traffic. However, web domains can be blocked at the DNS level, C&C web servers can be located and seized and the botmaster can be traced.

To adapt, botmasters came up with two major innovations. Bots are no longer hardcoded with a web address prior to deployment, but with a **Domain Generation Algorithm (DGA)** that takes date and time as seed values to generate custom domain names at a very rapid rate. The rationale is that it is very costly and time-consuming for law enforcement to seize a large number of domains whereas the botmaster has to register only one to successfully rendezvous with his bots in a given time-window. Conficker-C generated 50,000 domain names daily, distributed over 116 Top Level Domains (TLDs) which proved nearly impossible to block [11]. However, DGAs can be reverse-engineered. Security researchers hijacked the Torpig botnet for a period of ten days by registering certain domains ahead of the botmasters [12].

The second innovation is **Domain Flux**: botmasters now link several hundreds of destination IP addresses with a single fully qualified domain name in a DNS record (e.g. www.domain.com). These IP addresses are swapped with very high frequency (as often as every 3 min), so that different parties connecting to the same domain within minutes of each other are redirected to different locations. Furthermore, destination IP addresses often themselves point to infected hosts which act as *proxies* for the botmaster. Yet another layer of confusion can be added into the equation by similarly concealing the Authoritative Name Servers for the domain within this constantly changing *fast flux* cloud.

The third major botnet C&C infrastructure, decentralized **P2P networks**, have been used by Conficker, Nugache and Storm botnets in 2006–2007. Bots maintain individual routing tables, and every bot actively participates in routing data in the network, making it very difficult to identify C&C servers. However, P2P-based bots also have weak points: for instance, to bootstrap entry into the P2P network, Phatbot uses Gnutella cache servers on the Internet and Nugache bots are hardcoded with a seed list of IP addresses, both of which are centralized points of failure [13]. Security researchers have been able to detect P2P traffic signatures, successfully crawl P2P networks to enumerate the botnet, and poison bot routing tables to disrupt the botnet. In a concerted takedown effort, Symantec researchers took down the ZeroAccess botnet by flooding routing advertisements that overwhelmed bot routing tables with invalid or *sinkhole* entries, isolating bots from each other and crippling the botnet [14].

Some botnets employ multiple solutions for robustness, for example, Conficker uses HTTP-based C&C in addition to its P2P protocol [11]. More recently botnets have begun experimenting with **esoteric C&C mechanisms**, including darknets, social media and cloud services. The Flashback Trojan retrieved instructions from a Twitter account [15]. Whitewell Trojan used Facebook as

a rendezvous point to redirect bots to the C&C server [16]. Trojan.IcoScript used webmail services like Yahoo Mail for C&C communications [17]. Makadocs Trojan [18] and Vernot [19] used Google Docs and Evernote respectively as proxies to the botmaster. The results have been mixed. Network administrators rarely block these services because they are ubiquitously used, and C&C traffic is therefore hard to distinguish. On the other hand, C&C channels are again centralized and companies like Twitter and Google are quick to crack down on them.

2.2 Bitcoin

Bitcoin may be visualized as a distributed database which tracks the ownership of virtual currency units (bitcoins). Bitcoins are not linked to users or accounts but to *addresses*. A Bitcoin address is simply a transformation on a public-key, whereas, the private-key is used to *spend* the bitcoins associated with that address. A *transaction* is a statement containing an input address, an output address, and the quantity to be transferred, digitally signed using the private-key associated with the input. More complex transactions may include multiple inputs and outputs. All inputs and outputs are created using scripts that define the conditions to claim the bitcoins.

Transactions are circulated over the Bitcoin network, a decentralized global P2P network. Users known as *miners* collect transactions and craft them into *blocks*, which are chained into a *blockchain* to maintain a cryptographically verifiable ordering of transactions. Miners compete to solve a proof-of-work puzzle to insert their block into the blockchain. New blocks are generated at the rate of approximately once every ten minutes. The double spending problem of digital currencies is overcome by replicating the blockchain at the network nodes and using a consensus protocol to ensure global consistency of state.

Trustlessness is fundamental to Bitcoin: Bitcoin was deliberately designed to resist the kind of centralization, monetary control, and oversight which restrict fiat currencies [6]. Users have some degree of anonymity¹ which may be enhanced using Tor and mixing services. The decentralized nature of the network and the proof-of-work puzzle ensures that transactions in the network cannot be easily regulated. Bitcoin can only be subverted if a malicious party in the network musters more computing power than the rest of the network combined.

3 ZombieCoin

Here we outline briefly how ZombieCoin works:

- (1) We assume the botmaster owns Bitcoin credentials, i.e. a key pair (sk, pk) . The public-key, pk , is hardcoded into the bot binary file prior to deployment, so that bots can authenticate communication from the botmaster. Bots are also equipped with an instruction set to decode commands. Our implementation, described in Sect. 4, consists of simple instructions such as REGISTER, PING, UPDATE, etc. with associated parameters.

¹ Bitcoin technically provides *pseudonymity*, a weaker form of anonymity, in that Bitcoin addresses are not tied to identity and it is trivial to generate new addresses.

- (2) The botnet is then released into the wild. We assume there is an infection mechanism to propagate the botnet.
- (3) Bots then individually connect to the Bitcoin network and receive and propagate incoming transactions. All network communication proceeds as per the standard Bitcoin protocol specification described in [20]. By adhering to the standard protocol, the network behavior of the botnet to an outside observer is indistinguishable from the traffic of a genuine Bitcoin user.
- (4) The botmaster periodically issues C&C instructions by obfuscating and embedding them into transactions. Bots identify these transactions by scanning the *ScriptSig* field in the input which contains the botmaster's public-key, pk , and the digital signature (computed over the transaction) using private-key sk . Bots verify the signature and decode and execute the instructions.

3.1 Inserting C&C Instructions in Transactions

The most straightforward method is to insert C&C data in the **OP_RETURN output script function**. This function is a recent feature included in the 0.9.0 release of the Bitcoin Core client, allowing users to insert up to 40 bytes of data in transactions. This inclusion is due to immense lobbying by the Bitcoin community [21]. Developers anticipate the usage of this function to be along the lines of meaningful transaction identifiers (similar to text fields in online banking transactions), hash digests of some data such as contracts [22], cryptotokens, or even index values to link to other data stores. Analysis of a recent 80-block portion of the blockchain reveals that the OP_RETURN field was used in about a quarter of transactions in that portion [23], indicating that this feature is proving popular. One company has already launched timestamping services which rely on embedding hash data in this field [23].

This bandwidth is more than sufficient to embed most botnet commands which are typically instruction sets in the format $< command >< parameter > \dots < parameter >$. For instance, the DDoS attack library for Agobot [9] contains commands: *ddos.synflood* $< host >< time >< delay >< port >$ and *ddos.httpflood* $< url >< number >< referrer >< delay >< recursive >$, etc. Agobot has over ninety such commands and they can be encoded numerically using efficient schemes like Huffman coding to fit within the 40 byte limit.

A second approach offering greater bandwidth possibilities is to embed C&C instructions as **unspendable outputs**. This is a common technique and used by Counterparty [24] and Mastercoin [25]. We dissect a typical Mastercoin transaction in Fig. 1. The first output address, *1EXoDusjG...*, referred to as the Mastercoin Exodus Address, identifies this as a Mastercoin transaction. The last output address is an unspendable output, which decodes into a Mastercoin transaction. Very small bitcoin values are generally associated with such outputs because they cannot be redeemed. Up to 20 bytes of data may be inserted into an unspendable output, and a single transaction may have multiple such outputs. Proof of Existence [26], a Bitcoin-based notary public service, timestamps data by inserting hash digests as multiple unspendable outputs in transactions.

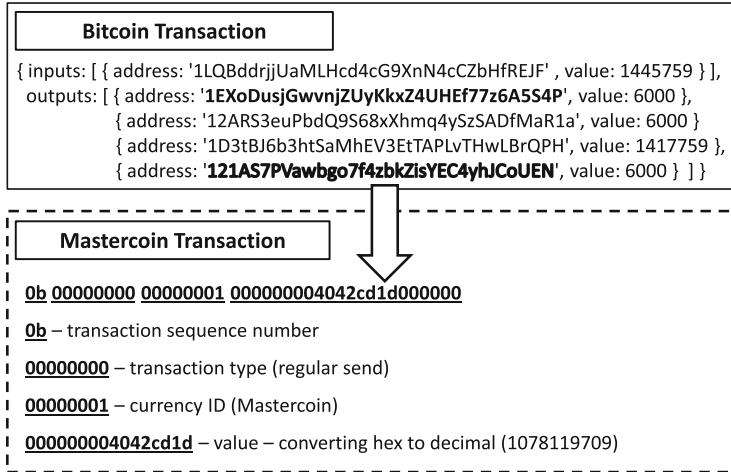


Fig. 1. Decoding a mastercoin transaction

Incidentally, however, Mastercoin, Counterparty, and Proof of Existence plan to migrate to using the OP_RETURN function [21]. As we noted, unspendable outputs are inherently wasteful. This method is also clumsy: Bitcoin clients maintain a live inventory of unspent transaction outputs (*UTXO*) to efficiently verify validity of new transactions. Clients cannot identify malformed outputs, with the result that these addresses populate the *UTXO* data set indefinitely (since they are never spent), affecting the efficiency of the network as a whole.

A more elegant technique is to communicate C&C messages by **key leakage**. Signing two different messages using the same random factor in the ECDSA signature algorithm allows an observer to derive the signer's private-key. Such instances have already been observed in the blockchain, resulting in coin theft [27]. In this case, the botmaster frames the C&C instruction within a 32 byte ECDSA private-key (including padding with random data so that identical commands do not always yield the same private-key). This is followed an obfuscation technique to give the data enough randomness to function as a private-key. The public-key is then derived. The botmaster then signs two transactions using the same random factor allowing bots to derive the private key.

This approach is used by Commitcoin [28] to insert hash digests in transactions. Bitcoins are not wasted using this method, and bandwidth is up to 32 bytes per input. However, two transactions are needed to leak the instructions.

A more covert solution is to use **subliminal channels**. Simmons [29, 30] notably demonstrated that two parties can set up a secret communications channel in digital signature schemes. This is again done by exploiting the random factor used by the signing algorithm. The botmaster creates a C&C instruction bitstring of length x bits. He then repeatedly generates signatures on the transaction using different random factors, until he gets a match, i.e. a signature, the first x bits of which match the target bitstring. He attaches this signature

to the transaction and publishes it. Nodes receive the transaction, verify that the signature is valid, and propagate it. Bots, on the other hand, extract the instructions from the first x bits and execute them.

Bandwidth is restricted using this technique due to the one-way nature of the signing function. Generating x bits of an ECDSA signature to match a bitstring takes on average 2^x iterations. For larger instructions, the botmaster may choose to split them into smaller target bitstrings inserted in multiple signatures. We briefly investigate here the practicality of this approach. We use an Intel i7 machine operating at 2.8 GHz with 8 GB RAM, running 64-bit Windows 7, and we use the OpenSSL toolkit to construct ECDSA signatures with subliminal channels of incrementing size. In each run we construct eight signatures matching a target string and record the time taken. Results are plotted in Fig. 2.

As demonstrated, it takes under 10 min (600 s) to *sequentially* generate eight signatures with subliminal channels of size 14 bits each. Total bandwidth in this case is 14.8 bits (i.e. 14 bytes overall). We consider here a couple of optimizations: first, we use *multithreading* to parallelize the operations across the multiple processors of the machine. It now takes about 3 min to generate eight signatures with 14-bit channels, a reduction of nearly 65 %.

Second, instead of passing each thread a single target bitstring, each thread now searches across the whole range of targets. The process stops as soon as each thread has located at least one distinct target. This *shared-search* step exploits the randomness of the signature generation process, increasing the odds of a successful match. We note an approximate 20 % improvement over the basic multithreading scenario, taking only about 2 min to generate eight 14-bit subliminal channels, which is very practical. The botmaster can order the resulting signatures accordingly in the transaction to construct the full channel.

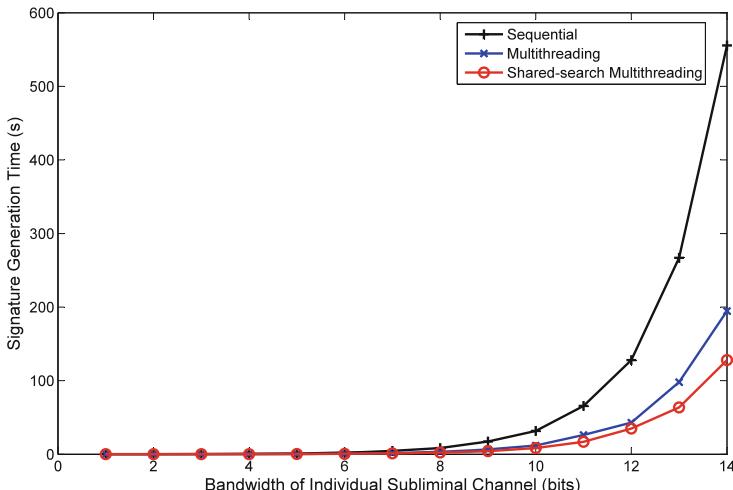


Fig. 2. Bandwidth vs. signature generation time for subliminal channels

4 Proof of Concept

We build a 14 node botnet and evaluate its performance over the Bitcoin network. We use the BitcoinJ library [31], which is an open source Java implementation of the Bitcoin protocol. We chose the Simplified Payment Verification (SPV) mode [32], which has a very low memory and traffic footprint, ideally suited for botnets. As opposed to Core nodes, SPV nodes do not replicate the entire blockchain but only a subset of block headers and filter incoming traffic to transactions of interest. Our bot application is 7 MB in size, the locally stored blockchain content is maintained at 626 kB, and at the network level, the bot's traffic is indistinguishable from that of any other SPV client.

To simulate a distributed presence, we installed our bots in multiple locations in the United States, Europe, Brazil, and East Asia using Microsoft's Azure cloud platform [33], and ran two bots locally in our Computing Science Department. The bots individually connect to the Bitcoin network, download peer lists, and scan for transactions circulated by the botmaster (us).

Our experiment loops approximately once per hour through an automated cycle of rudimentary instructions in the sequence depicted in Fig. 3. We embed C&C instructions in the OP_RETURN field and in (3-bit) subliminal channels in the authorized signatures. Bots are hardcoded with the a public-key, enabling them to identify our transactions. Bots receive transactions, verify, decode, and execute them.

We simulate botnet leasing in Step 3. Botmasters commonly monetize their botnet by partitioning and leasing it as “botnets for hire”. In our case, botmaster and tenant sign and publish a multi-input transaction containing the LEASE command. This transaction is a bona fide contract between botmaster and tenant and includes the lease payment in bitcoins from the tenant to botmaster. Bots verify input signatures, record the tenant's public-key, and accept C&C instructions issued by the tenant for the specified lease period.

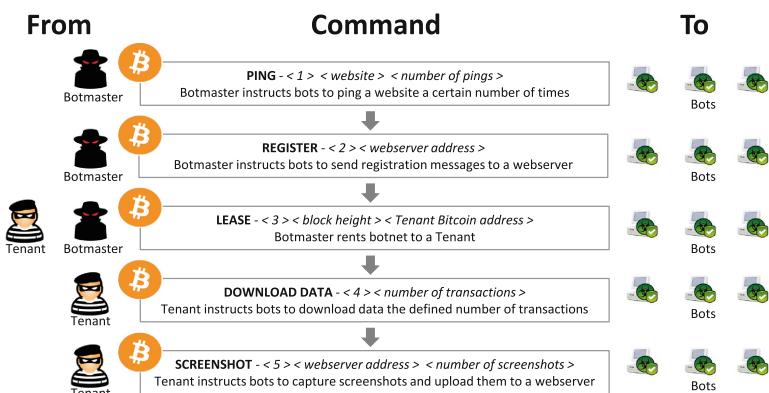


Fig. 3. Sequence of commands in the experiment

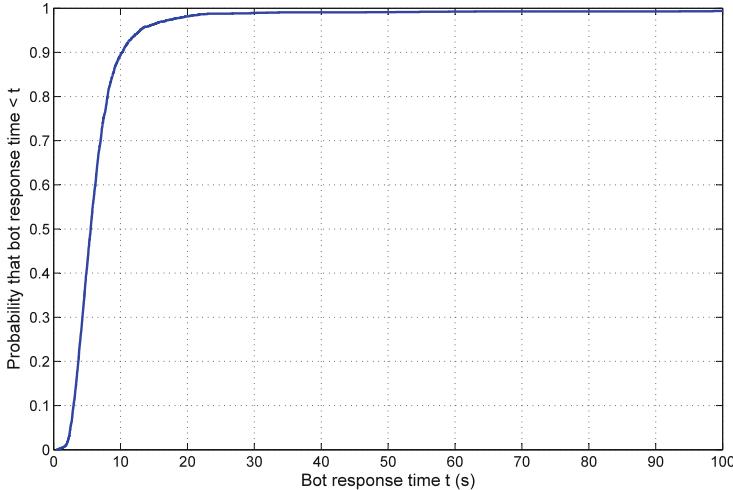


Fig. 4. Cumulative probability distribution of bot response time

When the tenant assumes control, he may send bots new encryption credentials or software modules. We simulate this with the DOWNLOAD command which uses transaction chaining to send bots a 256 byte RSA public-key, split over 7 back-to-back transactions. When bots receive the SCREENSHOT command, they capture a snapshot of the victim’s desktop, encrypt it using the tenant’s RSA public-key and send it to the web address specified.

We collect over 2300 responses from our bots over a 24 hour period. We are interested in the C&C channel latency and in the time it takes for bots to respond to an instruction. To synchronize readings over multiple time zones, we configure bots to set their clocks using a common timeserver.

Figure 4 plots the cumulative probability distribution of the bot response time. About 50 % of the time, the bots responded within 5 s, and 90 % of the time within 10 s. Median response time is 5.54s. In the interest of improved visualization, our results do not show outliers beyond the 100s mark. Only in 15 instances (0.6 % of overall results) was bot response time greater, ranging from 100–260 s.

5 Discussion

We believe our preliminary results highlight the realistic and practical aspects of ZombieCoin and we should take seriously the threat of botnets upgrading C&C communications onto the Bitcoin network.

So far we have assumed bots identify messages from the botmaster based on transaction input which raises the possibility of blacklisting the botmaster’s Bitcoin address. This is not likely to resolve the problem. For one, it would be

a form of regulation, a fundamental violation of the Bitcoin ethos [6], and we expect Bitcoin users would be the first to vigorously resist such attempts.

Second, such a step would require a significant protocol upgrade which could potentially degrade performance and usability of Bitcoin for legitimate users. Miners by themselves could, with relative ease, cooperate and ensure ZombieCoin transactions are ignored and do not appear in the blockchain. However, this does not solve the underlying problem of the circulation of valid ZombieCoin transactions throughout the network. In the current protocol version, nodes that receive incoming transactions perform checks for correctness (i.e. the input address is valid, the transaction is in the correct format, sum of inputs equals outputs, digital signature verification, etc.) and then forward the transaction on to other nodes. In our demo described earlier, our bots do not look up transactions from incoming blocks of the blockchain at approximate 10 min intervals, but receive them within a 5–12 s window as the transactions propagate throughout the network. Even if all C&C transactions are ultimately rejected by miners, the bots have already received them, validated them, and carried out the embedded instructions. Halting the propagation of these transactions in the Bitcoin network would require the cooperation of the majority of nodes in the network.

Furthermore, the botmaster can switch to alternate authentication strategies which do not rely solely on Bitcoin addresses but may use subliminal channels in transaction outputs or digital signatures. Botmasters could potentially keep escalating the fight, making it harder for legitimate clients to use the network.

In theory, an anti-virus installed on a victim’s machine could scan the Bitcoin network in lockstep with bots and block incoming C&C instructions. However, new malware are adept at evading anti-viruses: Torpig bots [12] contain rootkit functionality, executing their code prior to loading the OS, or injecting their code into legitimate processes to escape detection.

We would also make brief mention here of the costs of running ZombieCoin. Typically it costs about 3 cents (0.1mBTC) for every 1000 bytes of data in the transaction. If a botmaster were to issue one command every 20 min, this would translate to about US\$ 2.2 a day. Our experiment ran over 24 hours and 250 C&C instructions were sent at a cost of US\$ 7.50. This cost is trivial not only compared to the profits of successful botnets which is typically in the hundreds of thousands of dollars, but also if one considers the alternative scenario where a botmaster runs his own customized C&C network. This dramatically increases the odds of detection, botnet takedown, and risks exposing the botmaster.

Thus far we have not found any recognition of this threat among the Bitcoin community. We urgently need constructive dialogue regarding the grave risks associated with unregulated networks. Perhaps we also need to shift research focus back to traffic analysis and malware detection techniques. The new paradigm of software-defined networking (SDN) may hold some promise: there is already research suggesting SDN assists significantly in detecting malware-related anomalies at the network level [34].

We would stress here an earlier suggestion from the literature [12]: researchers and law enforcement should cultivate working relationships with registrars and ISPs to enable rapid response time to malware threats. Another approach

proposed before, but, to the best of our knowledge, never applied in practice is to combat the botnet problem at its root, the economy that drives it. Ford et al. [35] propose deliberately infecting large numbers of decoy virtual machines (honeypots) to join the botnet but remain under control of the white hats. By disruptive, unpredictable behavior, these sybils will actively undermine the economic relationship between botmaster and clients. An ad master for instance, may pay for a certain number of ad impressions, and sibyls making artificial clicks will not translate to the expected increase in actual sales. Targeting the economic incentive may prove a potent counter to the botnet threat.

6 Prior Work

Botnet-related research follows multiple strands. There are studies on the botnet economy [35–37]. Researchers have autopsied botnets, including early varieties like Agobot, SDBot [9], and state-of-the-art worms, Conficker [38], Storm [39], Waladec [40], and ZeroAccess [41]. There is extensive work on botnet tracking methods [42, 43] and traffic analysis and detection tools such as BotSniffer [44], BotMiner [45], and BotHunter [46]. Researchers have infiltrated botnets [12] and documented insider perspectives [47]. Readers interested in comprehensive surveys of the botnet phenomenon are directed to [48, 49].

There is a growing literature on exploring novel C&C mechanisms so that preemptive solutions may be devised. We summarize here a few such efforts:

Starnberg et al. present Overbot [50] which uses the P2P protocol Kademia for stealth C&C communications. The authors share our design concerns that bot traffic is covert and not easily distinguishable. However, there are critical differences: Overbot nodes carry the private key of the botmaster, and capturing one bot compromises the entire botnet’s communications. Furthermore, unlike our case where instructions are circulated within seconds, for Overbot this may take up to 12 hours. ZombieCoin also requires substantially less network management as the Bitcoin network handles message routing and global consistency.

The work closest to ours is that of Nappa et al. [51] who propose a C&C channel overlaid on the Skype network. Skype is closed-source, has a large user base, is resilient to failure, enforces default encryption, and is notoriously difficult to reverse engineer, all of which are ideal qualities for C&C communications. As in our case, disrupting this botnet would significantly impact legitimate Skype users. However, unlike Bitcoin, Skype is not designed to maintain low latency global consistency of state. Furthermore, after the Microsoft takeover in 2011, Skype has switched to a centralized cloud-based architecture [52].

Researchers have also proposed esoteric C&C mechanisms: Stegobot [53] creates subliminal channels on social networks by steganographic manipulation of user-shared images. Zeng et al. [54] describe a mobile P2P botnet concealing C&C communication in SMS spam messages. Desimone et al. [55] suggest creating covert channels in BitTorrent protocol messages. These solutions present interesting possibilities but are not very practical, with limitations in terms of bandwidth, latency and security.

7 Conclusion

In this paper we have described ZombieCoin, a mechanism to control botnets using Bitcoin. ZombieCoin inherits key strengths of the Bitcoin network, namely it is distributed, has low latency, and it would be hard to censor C&C instructions inserted in transactions without significantly impacting legitimate Bitcoin users. Our prototype implementation demonstrates that it is easy to build this C&C functionality by modifying freely available software, and experimental results show that instructions propagate in near real-time on the Bitcoin network.

We believe ZombieCoin poses a credible threat and we hope our work prompts further discussion and a step towards devising effective countermeasures.

Acknowledgements. This work is supported by the European Research Council (ERC) Starting Grant (No. 106591). The authors thank Hassaan Bashir, Mike Hearn, Paweł Widera, and Siamak Shahandashti for invaluable assistance with experiments and helpful comments.

References

1. Weber, T.: Criminals ‘may overwhelm the web’. BBC Home, 25 January 2007. Accessed on 22 July 2014
2. Dittrich, D.: So you want to take over a botnet. In: Proceedings of the 5th USENIX Conference on Large-Scale Exploits and Emergent Threats, pp. 6–6. USENIX Association (2012)
3. Stevenson, A.: Botnets infecting 18 systems per second, warns FBI. V3.co.uk, 16 July 2014. Accessed on 22 July 2014
4. Android smartphones ‘used for botnet’, researchers say 5 July 2012. <http://www.bbc.co.uk/news/technology-18720565>
5. Vincent, J.: Could your fridge send you spam? security researchers report ‘internet of things’ botnet. The Independent, 20 January 2014. Accessed on 22 July 2014
6. Bustillos, M.: The Bitcoin Boom. The New Yorker, April 2013. Accessed on 22 July 2014
7. Young, A., Yung, M.: Malicious Cryptography: Exposing Cryptovirology. John Wiley & Sons, Chichester (2004)
8. ICT-FORWARD Consortium. FORWARD: Managing Emerging Threats in ICT Infrastructures, 2007–2008. Accessed on 22 July 2014
9. Barford, P., Yegneswaran, V.: An inside look at botnets. In: Christodorescu, M., Jha, S., Maughan, D., Song, D., Wang, C. (eds.) Malware Detection. Advances in Information Security, vol. 27, pp. 171–191. Springer, New York (2007)
10. Westervelt, R.: Botnet Masters Turn to Google, Social Networks to Avoid Detection. TechTarget, 10 November 2009. Accessed on 4 Aug 2014
11. Bowden, M.: Worm: The First Digital World War. Atlantic Monthly Press, New York (2011)
12. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your botnet is my botnet: analysis of a botnet takeover. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), pp. 635–647. ACM (2009)

13. Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer-to-peer botnet. *IEEE Trans. Dependable Sec. Comput.* **7**(2), 113–127 (2010)
14. Neville, A., Gibb, R.: Security response: zeroaccess indepth. White paper, Symantec, 4 October 2013
15. Prince, B.: Flashback botnet updated to include twitter as C&C. SecurityWeek, 30 April 2012. Accessed on 22 July 2014
16. Lelli, A.: Trojan.Whitewell: What's your (bot) Facebook Status Today? Symantec Security Response Blog, October 2009. <http://www.symantec.com/connect/blogs/trojanwhitewell-what-s-your-bot-facebook-status-today>. Accessed on 22 July 2014
17. Kovacs, E.: RAT Abuses Yahoo Mail for C&C Communications. SecurityWeek, 4 August 2014. Accessed on 4 August 2014
18. Katsuki, T.: Malware Targeting Windows 8 Uses Google Docs. Symantec Official Blog, 16 November 2012. Accessed on 4 August 2014
19. Gallagher, S.: Evernote: So useful, even malware loves it. Ars Technica, 27 March 2013. Accessed on 4 August 2014
20. Protocol Specification. Bitcoin Wiki. Accessed 22 July 2014
21. Apodaca, R.L.: OP_RETURN and the Future of Bitcoin. Bitzuma, 29 July 2014. Accessed on 4 August 2014
22. Andresen, G.: Core Development Update #5. Bitcoin Foundation, 24 October 2013. Accessed on 4 Aug 2014
23. Bradbury, D.: BlockSign Utilises Block Chain to Verify Signed Contracts. CoinDesk, 27 August 2014. Accessed on 27 August 2014
24. Counterparty: Pioneering Peer-to-Peer Finance. Accessed on 22 July 2014
25. Willet, J.R.: The Second Bitcoin Whitepaper, v. 0.5, January 2012. <https://sites.google.com/site/2ndbtcwpaper/2ndBitcoinWhitepaper.pdf>. Accessed on 22 July 2014
26. Kirk, J.: Could the Bitcoin Network be Used as an Ultrasecure Notary Service? PCWorld, 24 May 2013. Accessed on 27 August 2014
27. Bos, J.W., Halderman, J.A., Heninger, N., Moore, J., Naehrig, M., Wustrow, E.: Elliptic curve cryptography in practice. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 156–174. Springer, Heidelberg (2014). IACR Cryptology ePrint Archive
28. Clark, J., Essex, A.: CommitCoin: carbon dating commitments with bitcoin. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 390–398. Springer, Heidelberg (2012)
29. Simmons, G.J.: The prisoners problem and the subliminal channel. In: Chaum, D. (ed.) Advances in Cryptology, pp. 51–67. Springer, Cambridge (1984)
30. Simmons, G.J.: The subliminal channel and digital signatures. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 364–378. Springer, Heidelberg (1985)
31. BitcoinJ: A Java implementation of a Bitcoin client-only node. <https://code.google.com/p/bitcoinj/>
32. Nakamoto, S.: Bitcoin: A Peer-to-peer Electronic Cash System (2009). <http://www.bitcoin.org/bitcoin.pdf>. Accessed on 22 July 2014
33. Azure: Microsoft's Cloud Platform. <https://azure.microsoft.com/en-gb/>
34. Mehdi, S.A., Khalid, J., Khayam, S.A.: Revisiting traffic anomaly detection using software defined networking. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 161–180. Springer, Heidelberg (2011)
35. Ford, R., Gordon, S.: Cent, five cent, ten cent, dollar: hitting botnets where it really hurts. In: Proceedings of the 2006 Workshop on New Security Paradigms, pp. 3–10. ACM (2006)

36. Franklin, J., Perrig, A., Paxson, V., Savage, S.: An inquiry into the nature and causes of the wealth of internet miscreants. In: ACM Conference on Computer and Communications Security, pp. 375–388 (2007)
37. Li, Z., Liao, Q., Striegel, A.: Botnet economics: uncertainty matters. In: Johnson, M.E. (ed.) Managing Information Risk and the Economics of Security, pp. 245–267. Springer, New York (2009)
38. Porras, P., Saidi, H., Yegneswaran, V.: A foray into confickers logic and rendezvous points. In: USENIX Workshop on Large-Scale Exploits and Emergent Threats (2009)
39. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.C.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), pp. 1–9 (2008)
40. Stock, B., Gobel, J., Engelberth, M., Freiling, F.C., Holz, T.: Walowdac-analysis of a peer-to-peer botnet. In: 2009 European Conference on Computer Network Defense (EC2ND), pp. 13–20. IEEE (2009)
41. Andriesse, D., Rossow, C., Stone-Gross, B., Plohmann, D., Bos, H.: Highly resilient peer-to-peer botnets are here: an analysis of gameover zeus. In: 2013 8th International Conference on Malicious and Unwanted Software: “The Americas” (MALWARE), pp. 116–123. IEEE (2013)
42. Cooke, E., Jahanian, F., McPherson, D.: The zombie roundup: understanding, detecting, and disrupting botnets. In: Proceedings of the USENIX SRUTI Workshop, vol. 39, p. 44 (2005)
43. Ramsbrock, D., Wang, X., Jiang, X.: A first step towards live botmaster traceback. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 59–77. Springer, Heidelberg (2008)
44. Gu, G., Zhang, J., Lee, W.: Botsniffer: detecting botnet command and control channels in network traffic. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS (2008)
45. Gu, G., Perdisci, R., Zhang, J., Lee, W. et al.: Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: USENIX Security Symposium, pp. 139–154 (2008)
46. Gu, G., Porras, P.A., Yegneswaran, V., Fong, M.W., Lee, W.: Bothunter: detecting malware infection through ids-driven dialog correlation. USENIX Secur. **7**, 1–16 (2007)
47. Cho, C.Y., Caballero, J., Grier, C., Paxson, V., Song, D.: Insights from the inside: a view of botnet management from infiltration. In: USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET) (2010)
48. Khattak, S., Ramay, N., Khan, K., Syed, A., Khayam, S.: A Taxonomy of Botnet Behavior, Detection, and Defense. IEEE Commun. Surv. Tutor. **16**(2), 898–924 (2014)
49. Silva, S.S.C., Silva, R.M.P., Pinto, R.C.G., Salles, R.M.: Botnets: a survey. Comput. Netw. **57**(2), 378–403 (2013)
50. Starnberger, G., Kruegel, C., Kirda, E.: Overbot: a botnet protocol based on kademlia. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm), p. 13. ACM (2008)
51. Nappa, A., Fattori, A., Balduzzi, M., Dell’Amico, M., Cavallaro, L.: Take a deep breath: a stealthy, resilient and cost-effective botnet using skype. In: Kreibich, C., Jahnke, M. (eds.) DIMVA 2010. LNCS, vol. 6201, pp. 81–100. Springer, Heidelberg (2010)

52. Whittaker, Z.: Skype ditched peer-to-peer supernodes for scalability, not surveillance 24 June 2013. <http://www.zdnet.com/skype-ditched-peer-to-peer-supernodes-for-scalability-not-surveillance-7000017215/>
53. Nagaraja, S., Houmansadr, A., Piyawongwisal, P., Singh, V., Agarwal, P., Borisov, N.: Stegobot: a covert social network botnet. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 299–313. Springer, Heidelberg (2011)
54. Zeng, Y., Shin, K.G., Hu, X.: Design of SMS commanded-and-controlled and P2P-structured mobile botnets. In: Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec), pp. 137–148 (2012)
55. Desimone, J., Johnson, D., Yuan, B., Lutz, P.: Covert channel in the bittorrent tracker protocol. In: International Conference on Security and Management. Rochester Institute of Technology (2012). <http://scholarworks.rit.edu/other/300>

Cuckoo Cycle: A Memory Bound Graph-Theoretic Proof-of-Work

John Tromp^(✉)

600 Route 25A, East Setauket, New York 11733, USA

john.tromp@gmail.com

<http://tromp.github.io/>

Abstract. We introduce the first graph-theoretic proof-of-work system, based on finding small cycles or other structures in large random graphs. Such problems are trivially verifiable and arbitrarily scalable, presumably requiring memory linear in graph size to solve efficiently. Our cycle finding algorithm uses one bit per edge, and up to one bit per node. Runtime is linear in graph size and dominated by random access latency, ideal properties for a memory bound proof-of-work. We exhibit two alternative algorithms that allow for a memory-time trade-off (TMTO)—decreased memory usage, by a factor k , coupled with increased runtime, by a factor $\Omega(k)$. The constant implied in $\Omega()$ gives a notion of memory-hardness, which is shown to be dependent on cycle length, guiding the latter’s choice. Our algorithms are shown to parallelize reasonably well.

1 Introduction

A *proof-of-work* (PoW) system allows a verifier to check with negligible effort that a prover has expended a large amount of computational effort. Originally introduced as a spam fighting measure, where the effort is the price paid by an email sender for demanding the recipient’s attention, they now form one of the cornerstones of crypto currencies.

As proof-of-work for new blocks of transactions, Bitcoin [1] adopted Adam Back’s hashcash [2]. Hashcash entails finding a nonce value such that application of a cryptographic hash function to this nonce and the rest of the block header, results in a number below a target threshold¹. The threshold is dynamically adjusted by the protocol so as to maintain an average block interval of 10 min.

Bitcoin’s choice of the simple and purely compute-bound SHA256 hash function allowed for an easy migration of hash computation from desktop processors (CPUs) to graphics-card processors (GPUs), to field-programmable gate arrays (FPGAs), and finally to custom designed chips (ASICs), with huge improvements in energy-efficiency at every step.

Since Bitcoin, many other crypto-currencies have adopted hashcash, with various choices of underlying hash function. the most well-known being *scrypt* as introduced by Tenebrix [3] (since faded into obscurity) and copied by Litecoin [4].

¹ or, less accurately, results in many leading zeroes.

Scrypt, designed as a sequential memory-hard key derivation function, was specifically chosen to resist the migration away from CPUs and be “GPU-hostile”. However, to adapt to the efficient verifiability requirement of proof-of-work, its memory footprint was severely limited, and migration slowed down only slightly.

Primecoin [5] introduced the notion of a number-theoretic proof-of-work, thereby offering the first alternative to hashcash among crypto-currencies. Primecoin identifies long chains of nearly doubled prime numbers, constrained by a certain relation to the block header. Verification of these chains, while very slow compared to bitcoin’s, is much faster than attempting to find one. This asymmetry between proof attempt and verification is typical in non-hashcash proofs of work. Recently, two other prime-number based crypto-currencies were introduced. Riecoin is based on finding clusters of prime numbers, and Gapcoin on finding large gaps between consecutive prime numbers.

Momentum [6] proposes finding birthday collisions of hash outputs, in what could well be the simplest possible asymmetric proof-of-work, combining scalable memory usage with trivial verifiability. In Sect. 12 we show that Momentum is in essence a special case of Cuckoo Cycle, one that is particularly susceptible to time-memory trade-offs.

Adam Back [7] has a good overview of proof-of-work papers past and present.

2 Motivation

Cuckoo Cycle aims to be an “egalitarian” proof-of-work, that is, to minimize performance-per-dollar differences across hardware architectures, and make mining—the process of looking for proofs—on commodity hardware cost-effective. This is to be achieved by making main memory latency a bottleneck, since DRAM latencies have remained relatively stable while cpu-speed and memory bandwidth vary highly across hardware architecture and process technology.

Our aim of a memory bound PoW translates to the following desirable properties:

MB1 a target memory footprint that exceeds a single memory chip

MB2 a pattern of necessarily random memory accesses

MB3 minimal computation per random memory access²

MB4 no feasible trade-off of memory for time³

A memory bound PoW aims to take advantage of the huge economies of scale of commodity DRAM production to make DRAM chips the most cost-effective vehicle for mining. Just as SRAM is one order of magnitude faster but two orders more expensive than DRAM, so it is conceivable that development and production of custom memory chips for a memory bound PoW will incur a sufficient cost premium to wipe out any performance gains.

² Preferably less than the roughly 50 nanosecond row activation delay for switching rows on a memory bank.

³ Rather arbitrarily defined as incurring an order of magnitude increase in time \times memory used per expected solution.

We thus disagree with the premise of [8] that PoWs should be compute bound in order to have ongoing energy costs dominate mining, which results in an expensive ASIC design arms race to drive up performance per Watt, rapid hardware obsolescence, and geographical centralization towards cheap electric power.

We will not strive for provable lower bounds on memory usage. Such bounds appear to be attainable only under the so-called *random oracle model*, where memory tends to be used merely as a store for chains of compute-intensive hash outputs. Instead, we present an efficient proof-finding algorithm along with our best attempts at memory-time trade-offs, and conjecture that these cannot be significantly improved upon. Lacking precise definitions and proofs of memory bound-ness, this work should be considered more empirical than formal.

3 Graph-Theoretic Proofs-of-work

We propose to base proofs-of-work on finding certain subgraphs in large pseudo-random graphs. In the Erdős-Rényi model, denoted $G(N, M)$, a graph is chosen uniformly at random from the collection of all graphs with N nodes and M edges. Instead, we choose edges deterministically from the output of a keyed hash function, whose key could be chosen uniformly at random. For a well-behaved hash function, these two classes of random graphs should have nearly identical properties.

Formally, fix a keyed hash function $h : \{0, 1\}^K \times \{0, 1\}^{W_i} \rightarrow \{0, 1\}^{W_o}$, and a small graph H as a target subgraph⁴. Now pick a large number $N \leq 2^{W_o}$ as the number of nodes, and $M \leq 2^{W_i-1}$ as the number of edges. Each key $k \in \{0, 1\}^K$ generates a graph $G_k = (V, E)$ where $V = \{v_0, \dots, v_{N-1}\}$, and

$$E = \{(v_{h(k, 2i) \bmod N}, v_{h(k, 2i+1) \bmod N}) | i \in [0, \dots, M-1]\} \quad (1)$$

The inputs $i \in [0, \dots, M-1]$ are also called *nonces*⁵. The graph has a *solution* if H occurs as a subgraph. Denote the number of edges in H as L . A proof of solution is an ordered list of L nonces that generate the edges of H 's occurrence in G_k . Such a proof is verifiable in time depending only on H (typically linear in L), independent of N and M .

A simple variation generates random bipartite graphs: $G_k = (V_0 \cup V_1, E)$ where (assuming N is even) $V_0 = \{v_0, v_2, \dots, v_{N-2}\}$, $V_1 = \{v_1, v_3, \dots, v_{N-1}\}$, and

$$E = \{(v_{2(h(k, 2i) \bmod \frac{N}{2})}, v_{2(h(k, 2i+1) \bmod \frac{N}{2})+1}) | i \in [0, \dots, M-1]\} \quad (2)$$

The expected number of occurrences of H as a subgraph of G is a function of both N and M , and in many cases is roughly a function of $\frac{M}{N}$ (half the average node degree). For fixed N , this function is monotonically increasing in M .

⁴ Hash functions generally have arbitrary length inputs, but here we fix the input width at W_i bits.

⁵ These *micro* nonces should be distinguished from the *macro* nonce used to generate key k .

To make the proof-of-work challenging, one chooses a value of M that yields less than one expected solution.

The simplest possible choice of subgraph is a fully connected one, or a *clique*. While an interesting choice, akin to the number-theoretic notion of a prime-cluster as used in Riecoin, we leave its consideration to a future paper.

4 Cuckoo Cycle

In this paper we focus on what is perhaps the next-simplest possible choice, the *cycle*. Specifically, we propose the hash function siphash with a $K = 128$ bit key, $W_i = W_o = 64$ input and output bits, $N \leq 2^{64}$ a 2-power, $M = N/2$, and H an L -cycle. Using the lightweight siphash2-4 with only 6 rounds helps to attain property MB3. The reason for calling the resulting proof-of-work Cuckoo Cycle is that inserting items in a Cuckoo hashtable naturally leads to cycle formation in random bipartite graphs.

5 Cuckoo Hashing

Introduced by Rasmus Pagh and Flemming Friche Rodler [9], a Cuckoo hashtable consists of two same-sized tables each with its own hash function mapping a key to a table location, providing two possible locations for each key. Upon insertion of a new key, if both locations are already occupied by keys, then one is kicked out and inserted in its alternate location, possibly displacing yet another key, repeating the process until either a vacant location is found, or some maximum number of iterations is reached. The latter is bound to happen once cycles have formed in the *Cuckoo graph*. This is a bipartite graph with a node for each location and an edge for every inserted key, connecting the two locations it can reside at. It matches the bipartite graph defined above if the cuckoo hashtable were based on function h . In fact, the insertion procedure suggests a simple algorithm for detecting cycles.

6 Cycle Detection in Cuckoo Cycle

We enumerate the M nonces, but instead of storing the nonce itself as a key in the Cuckoo hashtable, we store the alternate key location, and forget about the nonce. We thus maintain the *directed* cuckoo graph, in which the edge for a key is directed from the location where it resides to its alternate location. Moving a key to its alternate location thus corresponds to reversing its edge. The outdegree of every node in this graph is either 0 or 1. When there are no cycles yet, the graph is a *forest*, a disjoint union of trees. In each tree, all edges are directed, directly, or indirectly, to its *root*, the only node in the tree with outdegree 0. Initially there are just N singleton trees consisting of individual nodes which are all roots. Addition of a new key causes a cycle if and only if its two endpoints are nodes in the same tree, which we can test by following

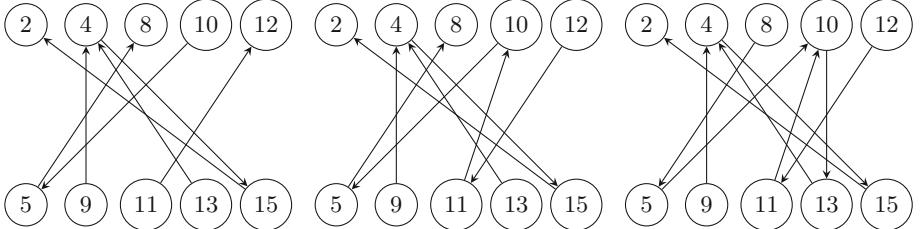


Fig. 1. Cycle formation and detection in a Cuckoo graph

the path from each endpoint to its root. In case of different roots, we reverse all edges on the shorter of the two paths, and finally create the edge for the new key itself, thereby joining the two trees into one. Let us illustrate this process with an actual example.

The left diagram in Fig. 1 shows the directed cuckoo graph for header “39” on $N = 8 + 8$ nodes after adding edges $(2, 15)$, $(4, 9)$, $(8, 5)$, $(4, 15)$, $(12, 11)$, $(10, 5)$ and $(4, 13)$ (nodes with no incident edges are omitted for clarity). In order to add the 8th edge $(10, 11)$, we follow the paths $10 \rightarrow 5 \rightarrow 8$ and $11 \rightarrow 12$ to find different roots 8 and 12. Since the latter path is shorter, we reverse it to $12 \rightarrow 11$ so we can add the new edge as $(11 \rightarrow 10)$, resulting in the middle diagram. In order to add to 9th edge $(10, 13)$ we now find the path from 10 to be the shorter one, so we reverse that and add the new edge as $(10 \rightarrow 13)$, resulting in the right diagram. When adding the 10th edge $(8, 9)$, we find the paths $8 \rightarrow 5 \rightarrow 10 \rightarrow 13 \rightarrow 4 \rightarrow 15 \rightarrow 2$ and $9 \rightarrow 4 \rightarrow 15 \rightarrow 2$ with equal roots. In this case, we can compute the length of the resulting cycle as 1 plus the sum of the path-lengths to the node where the two paths join. In the diagram, the paths join at node 4, and the cycle length is computed as $1 + 4 + 1 = 6$.

7 Union-Find

The above representation of the directed cuckoo graph is an example of a *disjoint-set data structure* [10], and our algorithm is closely related to the well-known union-find algorithm, where the find operation determines which subset an element is in, and the union operation joins two subsets into a single one. For each edge addition to the cuckoo graph we perform the equivalent of two find operations and one union operation. The difference is that the union-find algorithm is free to add directed edges between arbitrary elements. Thus it can join two subsets by adding an edge from one root to another, with no need to reverse any edges. Our algorithm on the other hand solves the union-find problem by maintaining a direction on all union operations while keeping the maximum outdegree at 1.

8 Cuckoo Cycle Basic Algorithm

The above algorithm for inserting edges and detecting cycles forms the basis for our basic proof-of-work algorithm. If a cycle of length L is found, then we solved the problem, and recover the proof by storing the cycle edges in a set and enumerating nonces once more to see which ones generate edges in the set. If a cycle of a different length is found, then we keep the graph acyclic by ignoring the edge. There is some risk of overlooking other L -cycles through that edge, but when the expected number of cycles is low (which is what we design for), this ignoring of cycle forming edges hardly affects the rate of solution finding.

This algorithm is available online at <https://github.com/tromp/cuckoo> as either the C-program simple_miner.cpp or the Java program SimpleMiner.java. A proof verifier is available as cuckoo.c or Cuckoo.java, while the repository also has a Makefile, as well as the latest version of this paper. ‘make example’ reproduces the example shown above. The simple program uses 32 bits per node to represent the directed cuckoo graph, plus about 64 KB per thread for two path-following arrays. The left plot in Fig. 2 shows both the total runtime in seconds and the runtime of just the hash computation, as a function of $(\log N)$. The latter is purely linear, while the former is superlinear due to increasing memory latency as the nodes no longer fit in cache. The right plot show this more clearly as the percentage of hashing to total runtime, ending up around 5 %.

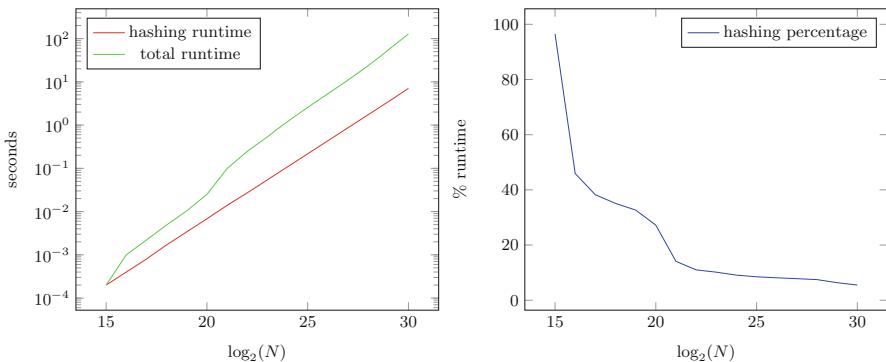


Fig. 2. Runtime and compute intensity of the basic algorithm

The left plot in Fig. 3 shows the probability of finding a 42-cycle as a function of the percentage edges/nodes, while the right plot shows the average number of memory reads and writes per edge as a function of the percentage of processed nonces (progress through main loop). Both were determined from 10000 runs at size 2^{20} ; results at size 2^{25} look almost identical. In total the basic algorithm averages 3.3 reads and 1.1 writes per edge.

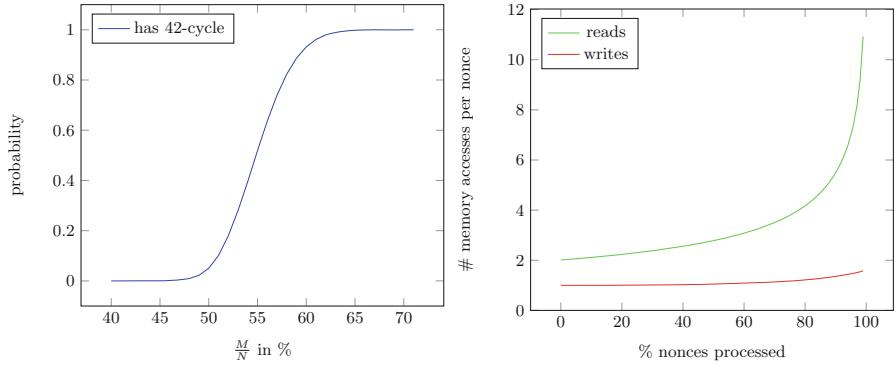


Fig. 3. Threshold nature of solution, and increasing memory usage on threshold approach

9 Difficulty Control

The ratio $\frac{M}{N}$ determines a base level of difficulty, which may suffice for applications where difficulty is to remain fixed. Ratios $\frac{M}{N} \geq 0.7$ are suitable when a practically guaranteed solution is desired.

For crypto currencies, where difficulty must scale in precisely controlled manner across a large range, adjusting the number of edges is not suitable. The implementation default $\frac{M}{N} = \frac{1}{2}$ gives a solution probability of roughly 2.2 %, while the average number of cycles found increases slowly with size; from 2 at 2^{20} to 3 at 2^{30} .

For further control, a difficulty target $0 < T < 2^{256}$ is introduced, and we impose the additional constraint that the sha256 digest of the cycle nonces in ascending order be less than T , thus reducing the success probability by a factor $\frac{2^{256}}{T}$.

10 Edge Trimming

David Andersen [11] suggested drastically reducing the number of edges our basic algorithm has to process, by repeatedly identifying nodes of degree one and eliminating their incident edge. Such *leaf edges* can never be part of a cycle. This works well when $\frac{M}{N} \leq \frac{1}{2}$ since the expected degree of a node is then at most 1, and a significant fraction of edges are expected to be leaf edges.

Trimming is implemented in our main algorithm in `cuckoo_miner` and `hcuckoo_miner.cpp`. It maintains a set of *alive* edges as a bit vector. Initially all edges are alive. In each of a given number of trimming rounds, it shrinks this set as follows. A vector of 2-bit degree counters, one per even node, is initialized to all zeroes. Next, for all alive edges, compute its even endpoint and increase the corresponding counter, capping the value at 2. Next, for all alive edges, compute its even endpoint and if the corresponding counter is less than 2, set the edge to

be not-alive. These steps, both of which cause the random accesses required in property MB2, are repeated for all odd endpoints.

Preprocessor symbol PART_BITS, whose value we'll denote as B , allows for *counter partitioning*, which trades off node counter storage for runtime, by processing nodes in multiple passes depending on the value of their B least significant bits⁶. The memory usage is M bits for the alive set and $N/2^B$ for the counters.

The diagrams in Fig. 4 show two rounds of edge trimming on the earlier example. In round one even nodes 2 and 12 lose their single incident edge and in round two, odd nodes 11 and 15 lose their remaining single incident edge. At this point only the 6-cycle is left, so further trimming would be pointless.

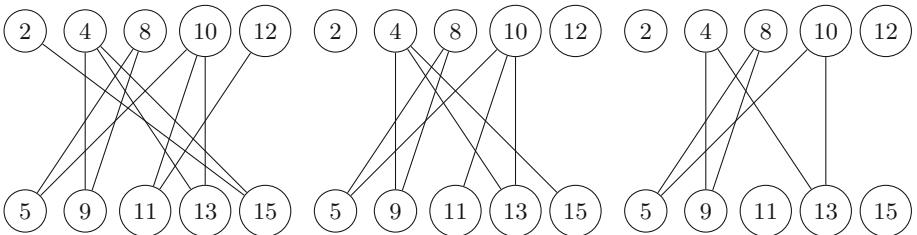


Fig. 4. Trimming of edges which cannot be part of a cycle

After all edge trimming rounds, the counter memory is freed, and allocated to a custom cuckoo_hashtable (based on [12]) that presents the same interface as the simple array in the basic algorithm, but gets by with much fewer locations, as long as its *load*, the ratio of remaining edges to number of locations, is bounded away from 1; e.g. under 90 percent.

The number of trimming rounds, which can be set with option `-n`, defaults to $1 + (B + 3) * (B + 4)/2$, which was determined empirically to achieve a load close to 50 %.

11 Time-Memory Trade-Offs (TMTOs)

David Andersen also suggested an alternative method of trimming that avoids storing a bit per edge. Expanding on that idea led to the algorithm implemented in `tomato_miner.h`, which, unlike the main algorithm, can trade-off memory directly for runtime. On the downside, to even achieve memory parity with the main algorithm, it already incurs a big slowdown. To the extent that this slowdown is unavoidable, it can be called the *memory hardness* of the proof-of-work.

The TMTO algorithm selects a suitably small subset Z of even vertices as a base layer, and on top of that builds a breadth-first-search (BFS) forest of

⁶ excluding the very least significant bit distinguishing even from odd nodes.

depth $L/2$, i.e. half the cycle length. For each new BFS layer, it enumerates all edges to see which ones are incident to the previous layer, adding the other endpoint. It maintains a directed forest on all BFS nodes, like the base algorithm does on all nodes. For increased efficiency, the base layer Z is filtered for nodes with multiple incident edges. If the graph has an L -cycle one of whose nodes is in Z , then the above procedure will find it. If one choice of Z doesn't yield a solution, then the data structures are cleared and the next subset is tried.

A variation on the above algorithm omits the filtering of Z , and expands the BFS to a whole L levels. This way, an L -cycle will be found as long as the distance from (any node in) Z to the cycle is at most $L/2$. It thus has a much higher chance of finding a cycle, but requires more space to store the significantly bigger BFS forest.

For each value of $L \in \{2, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64\}$ we ran these 2 algorithms on 200 graphs of size 2^{25} that include an L -cycle, choosing subset size as a 2-power that results in a memory usage of 4MB, and analysed the distribution of number of subsets tried before finding a solution. Since there is possible overlap between the BFS forests of different initial subsets, especially with the second algorithm, the distributions are skewed toward lower numbers. To maximize solution finding rate then, it pays to give up on a graph when the first few subsets tried fail to provide a solution. For each algorithm and cycle length, we determined the minimum number of tries needed to guarantee solutions in at least 50 of the 200 graphs. In Fig. 5 we plot the slowdown relative to the reference algorithm also using 4MB (2MB for edges and 2MB for nodes).

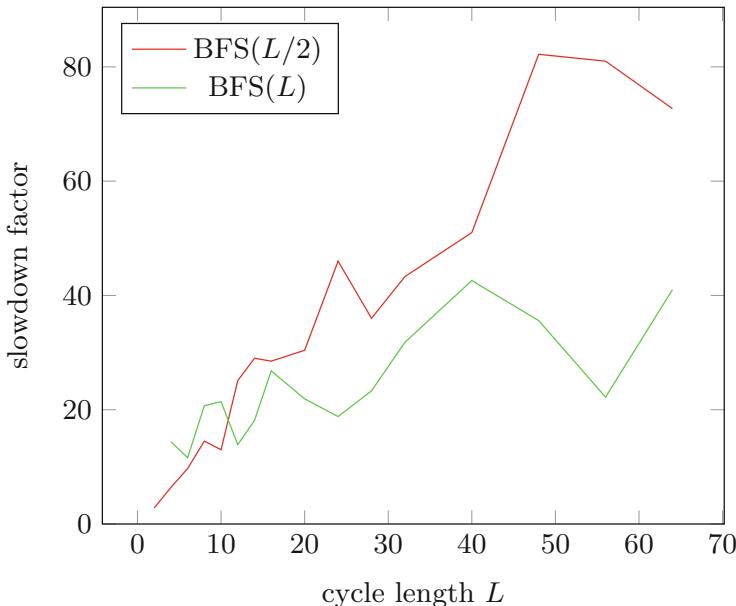


Fig. 5. Reduction in solution finding rate for two TMTO algorithms

The zigzagging is caused by the current implementation being limited to 2-power sizes of both subsets and cuckoo tables while the load of the latter is kept between 45 % and 90. The $\text{BFS}(L)$ algorithm exhibits at least one order of magnitude slowdown, that grows very slowly with cycle length, while the $\text{BFS}(L/2)$ algorithm exhibits roughly linear slowdown. Assuming that these algorithms cannot be significantly improved upon, this shows Cuckoo Cycle with larger cycle lengths satisfying property MB4.

12 Choice of Cycle Length

A cycle of length 2 means that two nonces produce identical edge endpoints—a *collision* in edge space. The Momentum proof-of-work looks for collisions on 50 bits of hash output among 2^{26} nonces. This is in essence Cuckoo Cycle with $N = 2^{25} + 2^{25}$ nodes and cycle length $L = 2$, with two differences.

First, edges are generated not by Eq. (2), but by splitting a SHA512 hash of $(k, \text{nonce}/8)$ into 8 64-bit words, taking the most significant 50 bits of the $(\text{nonce} \bmod 8)th$ one, and viewing that as a pair of two 25-bit edge endpoints, appending a bit to make them even and odd.

Second, the choice of $M = 2^{26}$ gives a ratio $\frac{M}{N}$ of 1 rather than $\frac{1}{2}$ and as such prohibits the use of edge trimming.

Since the extreme case of $L = 2$ is so special, there is likely to be a greater variety of algorithms that are more efficient than for the general case. While we haven’t found (and don’t know of) a improved main algorithm, we did find an improved $\text{BFS}(L/2)$ TMTQ algorithm (implemented in momentomatum.cpp) that cuts the memory usage in half, resulting in a slowdown of only 1.75—a lack of memory-hardness.

The preceding analysis suggests that cycle length should be at least 20 to guard against the more efficient $\text{BFS}(L/2)$ algorithm, with an additional safety factor of 2.

In order to keep proof size manageable, the cycle length should not be too large either. We thus consider 20-64 to be a healthy range, and suggest the use of the average of 42.

The plot below shows the distribution of cycle lengths found for sizes $2^{10}, 2^{15}, 2^{20}, 2^{25}$, as determined from 100000, 100000, 10000, and 10000 runs respectively. The tails of the distributions beyond $L = 100$ are not shown. For reference, the longest cycle found was of length 2120.

13 Parallelization

All our implementations allow the number of threads to be set with option `-t`. For $0 \leq t < T$, thread t processes all nonces $t \bmod T$. Parallelization in the basic algorithm presents some minor algorithmic challenges. Paths from an edge’s two endpoints are not well-defined when other edge additions and path reversals are still in progress. One example of such a path conflict is the check for duplicate edges yielding a false negative, if in between checking the two endpoints, another

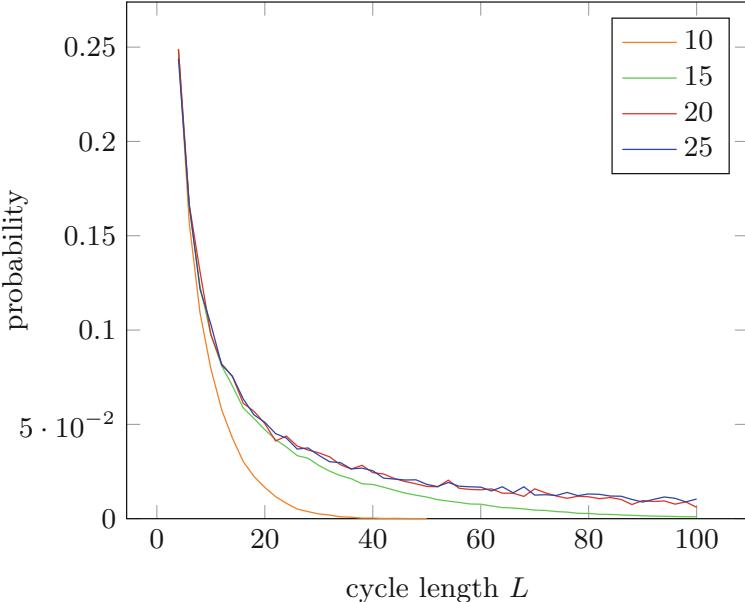


Fig. 6. Distribution of cycle lengths in random graphs

thread reverses a path through those nodes. Another is the inadvertent creation of cycles when a reversal in progress hampers another thread’s path following causing it to overlook root equality. Thus, in a parallel implementation, path following can no longer be assumed to terminate. Instead of using a cycle detection algorithm such as [13], our implementation notices when the path length exceeds MAXPATHLEN (8192 by default), and reports whether this is due to a path conflict (Fig. 6).

In the main algorithm, cycle detection only takes a small fraction of total runtime and the conflicts above could be avoided altogether by running the cycle detection single threaded.

In edge trimming, parallelization is achieved by partitioning the set of edges. To maintain efficient access to the bitmap of live edges, each thread handles words (of 32 edge-bits each) spaced T apart.

Atomic access is used by default for accessing the 2-bit counters. Disabling this results in a small chance of removing multiple edges incident to a node that access the counter at the same time.

The implementation further benefits from bucketing the addresses of counters to be updated or tested, based on their most significant bits. Thus, when a bucket becomes full and is emptied by actually performing those updates/tests, the accesses are limited to a certain address range, which turns out to reduce memory access latencies.

The plot below shows the speedup over single thread performance achieved by multithreading at 2^{30} nodes and various counter-partition levels.

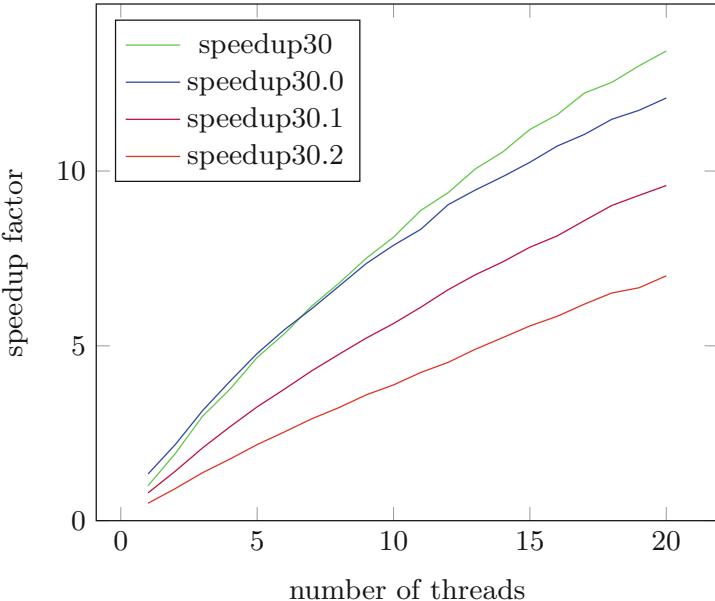


Fig. 7. Multi-threading speedup

14 Choice of Graph Size

For cryptocurrency purposes, the choice of Cuckoo graph size should be in accordance to its block interval time. To illustrate, suppose an average desktop machine needs 1 min for a single proof attempt, and the block interval time is only 2 min. Then it will waste a large fraction (almost half) of its attempts, as about half the time, someone else finds a proof in under 2 min. To reduce such waste to a small percentage, the time for a single proof attempt should be a similarly small fraction of the block interval time. This desirable property is known as *progress-freeness*, and in our case is achieved more easily with a small graph (and hence memory) size (Fig. 7).

Larger memory sizes have two advantages. Beyond satisfying property MB1, they also make it harder for botnets to mine without causing excessive swapping. Sending a computer into swap-hell will likely alert its owner and trigger a cleanup, so botnet operators can be expected to eschew memory bound PoWs in favor of low-memory ones.

We expect these opposing goals to lead to graph sizes from 2^{28} to 2^{32} , with the larger ones geared more toward longer block interval times and faster mining hardware.

15 Dynamic Sizing

Ideally, graph size should grow with evolving memory chip capacities, so as to preserve property MB1. Although these have shown remarkable adherence to Moore’s Law in the past, this cannot be relied on for the more distant future. We therefore propose to re-evaluate the graph size every so-many difficulty adjustments. If the difficulty target is sufficiently low, then the graph size is deemed to have become “too easy” for existing hardware, and gets doubled.

In order to make this transition smoother and avoid severe loss of proof-of-work power, we propose having a range of sizes allowed at any time, namely k consecutive 2-powers for some small number $k \geq 2$. As with Myriad-coin, separate difficulty controls are maintained for each size, adjusted so that each size accounts for roughly $\frac{1}{k}$ of all blocks.

Doubling graph sizes is then equivalent to disabling the smallest 2-power, and enabling a new largest one, whose initial difficulty target is twice that of the previous largest. Even if none of the hardware that was working on the smallest 2-power is repurposed for a larger size, since this hardware only accounted for a fraction $\frac{1}{k}$ of the rewards, the loss of proof-of-work power should be acceptable.

It remains to decide what exact form the “difficulties too low” condition should take.

16 Conclusion

Cuckoo Cycle is a novel graph-theoretic proof-of-work design that combines scalable memory requirements with instant verifiability, and the first where memory latency dominates the runtime.

Barring any unforeseen memory-time trade-offs, it makes for a near-ideal memory bound proof-of-work whose cost effectiveness on commodity hardware could greatly benefit decentralization of mining.

References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical Report, May 2009. <http://www.bitcoin.org/bitcoin.pdf>
2. Back, A.: Hashcash - a denial of service counter-measure. Technical Report, August 2002. (implementation released in Mar 1997)
3. Lolicust: [announce] tenebrix, a cpu-friendly, gpu-hostile cryptocurrency, September 2011. <https://bitcointalk.org/index.php?topic=45667.0>
4. Coblee: [ann] litecoin - a lite version of bitcoin. launched! October 2011. <https://bitcointalk.org/index.php?topic=47417.0>
5. King, S.: Primecoin: Cryptocurrency with prime number proof-of-work. Technical Report, July 2013. <http://primecoin.org/static/primecoin-paper.pdf>
6. Larimer, D.: Momentum - a memory-hard proof-of-work via finding birthday collisions. Technical Report, October 2013. www.hashcash.org/papers/momentum.pdf
7. Back, A.: Hashcash.org, February 2014. <http://www.hashcash.org/papers/>

8. Poelstra, A.: Asics and decentralization faq (2014). <https://download.wpsoftware.net/bitcoin/asic-faq.pdf>
9. Pagh, R., Rodler, F.F.: Cuckoo hashing. *J. Algorithms* **51**(2), 122–144 (2004). doi:[10.1016/j.jalgor.2003.12.002](https://doi.org/10.1016/j.jalgor.2003.12.002)
10. Wikipedia, Disjoint-set data structure – wikipedia, the free encyclopedia (2014). Accessed from 23-March-2014. http://en.wikipedia.org/w/index.php?title=Disjoint-set_data_structure
11. Andersen, D.: A public review of cuckoo cycle, April 2014. <http://da-data.blogspot.com/2014/03/a-public-review-of-cuckoo-cycle.html>
12. Preshing, J.: The world's simplest lock-free hash table, June 2013. <http://preshing.com/20130605/the-worlds-simplest-lock-free-hash-table/>
13. Brent, R.P.: An improved Monte Carlo factorization algorithm. *BIT* **20**, 176–184 (1980)

When Bitcoin Mining Pools Run Dry

A Game-Theoretic Analysis of the Long-Term Impact of Attacks Between Mining Pools

Aron Laszka¹(✉), Benjamin Johnson², and Jens Grossklags³

¹ Institute for Software Integrated Systems, Vanderbilt University, Nashville, USA
laszkaaron@gmail.com

² CyLab, Carnegie Mellon University, Pittsburgh, USA

³ College of Information Sciences and Technology,
Pennsylvania State University, University Park, USA

Abstract. Bitcoin has established itself as the most successful cryptocurrency with adoption seen in many commercial scenarios. While most stakeholders have jointly benefited from the growing importance of Bitcoin, conflicting interests continue to negatively impact the ecosystem. In particular, incentives to derive short-term profits from attacks on mining pools threaten the long-term viability of Bitcoin.

We develop a game-theoretic model that allows us to capture short-term as well as long-term impacts of attacks against mining pools. Using this model, we study the conditions under which the mining pools have no incentives to launch attacks against each other (i.e., *peaceful* equilibria), and the conditions under which one mining pool is marginalized by attacks (i.e., *one-sided attack* equilibria). Our results provide guidelines for ensuring that the Bitcoin ecosystem remains long-term viable and trustworthy.

Keywords: Bitcoin · Mining · Attacks · DDoS · Game theory

1 Introduction

Conceived in 2008, Bitcoin is a cryptocurrency system which is controlled through an online communication protocol and facilitated in a decentralized fashion [15]. Bitcoin has experienced considerable growth in popularity and constitutes the dominant cryptocurrency [2]. It has also increasingly found adoption as a viable payment scheme in mainstream electronic commerce.

Despite setbacks, such as the closure of the Mt. Gox exchange, most stakeholders of the Bitcoin ecosystem have profited from its development and expect to benefit also in the future from the trust placed in the cryptocurrency. As such, participants in the Bitcoin ecosystem share a *common goal* with the improvement (or avoidance of the erosion) of trust of the currency system. Interactions in most economic systems usually involve such common but at the same time also *conflicting interests* [18]. In the Bitcoin ecosystem, such misaligned incentives are manifested in several ways.

Most centrally, the process of mining new bitcoins is organized in the form of a race in which the miner that solves a proof-of-work task first will be rewarded; all other miners will leave empty-handed. Mining involves a probabilistic element, so that not only the most powerful miner would win a particular round of this competition with certainty. Nevertheless, individual miners have found it beneficial to join forces in the form of *mining pools*. For example, averaging mining proceeds across many participants makes earnings more predictable. The specific setup of each mining pool typically differs across several dimensions which can be tangible (e.g., related to the computing and communication infrastructure) or intangible (such as reputation or details of the payout schemes). We term the sum of these factors the *attractiveness* of a mining pool.

However, the decentralized and quasi-anonymous nature of the Bitcoin ecosystem also lowers the bar for unfair competition in the form of different *attacks* which can benefit a malicious mining pool. First, attackers may abuse the resources of unsuspecting computer users for mining purposes through security compromises [8, 16]. Second, attackers may attempt to redirect or siphon off mining capabilities from a competing pool [12]. Third, attackers may diminish the mining power of competing pools, for example, through Distributed Denial of Service (DDoS) attacks [20], or by exploiting specific weaknesses in the implementation of the procedure/software used by a particular pool.

The third dimension of unfair competition has been the focus of two recent research contributions. Vasek et al. provided empirical evidence that, during a two-year period, about 29 % of all known mining pools had been the subject of at least one DDoS attack, and for mining pools above 5 % share of hash rate, the likelihood of suffering an attack was 63 %. They further characterized the types of pools that are more likely to be attacked [20]. Building on these findings, we developed a game-theoretic model in [9], which investigated the adversarial interaction between two representative mining pools that can choose between productive and destructive investments (i.e., computing power vs. DDoS attack on its competitor). We found that the relative size of the mining pools is a critical factor for the incentives to engage in attacks.

Both research studies primarily focus on the immediate impact of attacks on mining pools, i.e., the temporary shutdown of mining power and its payoff consequences. However, previous research on DDoS attacks in the context of electronic commerce has shown that the future (medium or long-term) impact of service unavailability can actually be more significant [6]. In the context of mining pools, individual members can permanently shift to an unaffected pool, which lowers the future prospects of the attacked pool. Even though the long-term impact of attacks can have a strong influence on the behavior of service providers, this phenomenon has not been studied from a theoretical perspective for DDoS attacks in general, or in the context of Bitcoin mining pools in particular [9].

In this paper, we develop a game-theoretic model that allows us to investigate the long-term impact of attacks against mining pools. Using this model, we study the conditions under which the mining pools have no incentives to launch attacks

against each other (i.e., *peaceful* equilibria), and the conditions under which one mining pool is marginalized by attacks (i.e., *one-sided attack* equilibria). Our results provide guidelines for ensuring that the Bitcoin ecosystem remains long-term viable and trustworthy.

The remainder of this paper is organized as follows. In Sect. 2, we discuss related work relevant to the context of DDoS attacks in networked systems. We describe our model and key assumptions in Sect. 3. We conduct our analysis and present numerical results/illustrations in Sects. 4 and 5, respectively. We offer concluding remarks in Sect. 6.

2 Related Work

Decision-making in the context of security has been extensively studied using various game-theoretic approaches [10, 14]. Of particular interest to our work are studies which address the incentives for adversarial behaviors. For example, Schechter and Smith [17] build upon the literature on the economics of crime to construct a model of attackers in the computer-security context. The authors derive penalties and probabilities of enforcement that will deter a utility-optimizing attacker, who evaluates the risks and rewards of committing an offense. Clark and Konrad propose a game-theoretic model with one defender and one attacker [4]. In their model, the defending player has to successfully protect multiple nodes, while the attacker needs to compromise only a single node. Fultz and Grossklags study the competition between multiple strategic attackers in different interdependent decision-making scenarios [5, 7].

Previous economic work has improved our understanding of DDoS attacks and potential countermeasures. Focusing on defender behaviors, Christin et al. investigate the incentives of a group of bounded rational agents when they face the threat of being absorbed into a botnet, e.g., for the purpose of a DDoS attack [3]. In contrast, Liu et al. model attackers and work towards identifying DDoS attacker strategies in a specific case study [13]. Li et al. model the incentives of a botnet master to maintain a zombie network for the primary purpose of renting a sufficiently large subset to a DDoS attacker [11]. The authors investigate whether this business relationship can remain profitable if defenders can pollute the botnet with decoy machines (which lowers the effectiveness of a DDoS attack). In addition, there are several other research studies which are concerned with the organization of effective countermeasures against DDoS [19, 21].

As discussed in the introduction, our current work draws on Vasek et al. who provided empirical evidence on the prevalence of DDoS attacks in the Bitcoin economy [20]. They showed, for example, that the size of mining pools is related to the probability of being targeted by an attack. Those findings motivated us to develop a game-theoretic model of attack behaviors between two mining pools which is, however, restricted to studying short-term effects of attacks [9].

3 Model

3.1 Overview

Our modeling framework is designed to capture two distinct effects of attacks against mining pools. The first, and most obvious is a short-term effect on the revenue of the attacked pool. While an attack is ongoing, the communication of the pool is disrupted, and hence the revenue decreases. The second effect is a longer term decrease in the size of the pool. Due to the myopic behavior of miners, an ongoing attack may cause some miners to permanently leave the attacked pool and mine for other pools.

In our previous paper [9], we focused on the first effect and did not take into account the second. In this paper, we extend our analysis to incorporate both effects using a sequential game played over an indefinite number of rounds. In each round, the choices of players result in both short-term revenue consequences, which affect player utilities, as well as long-term migration consequences, which affect the relative sizes of pools in the next round.

Miner migration is an interesting feature in itself. Our modeling framework assumes that there is some level of migration in each round, regardless of any attacks. That is to say there is a percentage of miners who are sufficiently fluid in their preferences that they re-evaluate their choice of pool each round. The remaining percentage of miners in a given round will continue mining for the same pool in the next round.

Our main focus in studying this model will be determining steady-state equilibrium strategies. These are strategies that stabilize the long-term migration effects, so that the players' sizes remain the same from round to round; and that also constitute best-response strategies for each player. Steady-state equilibria are consistent with what we observe in the Bitcoin ecosystem, where we observe little change in the relative sizes of pools from round to round.¹

Table 1 summarizes the notations used in the model.

3.2 Players

Our game has exactly two players: a bigger mining pool B and a smaller mining pool S . Each pool has a base level of attractiveness which we parameterize with two constants A_B, A_S . We may interpret A_B (for example) as the percentage of fluid miners who will migrate to pool B in the next round.

In contrast to the attractiveness levels which are fixed for the duration of the game, each pool also has a current size for each round. For example, $s_B^{(k)}$ is the relative size of pool B in round k . Relative size is interpreted to mean the percentage of hash power possessed by its miners, compared to the entire Bitcoin ecosystem.

¹ Steady-state equilibrium analysis has been used relatively sparingly in the security economics literature (see, for example, [1]), while it is a frequently employed solution concept in other areas of economics.

Table 1. Table of Notations

Symbol(s)	Constraints	Description
M	$\in [0, 1]$	Base miner migration rate
C	$\in [0, 1]$	Unit cost of attack
A_B, A_S	$\in [0, 1]$ and $A_B + A_S \in [0, 1]$	Relative attractiveness of the pool
$s_B^{(k)}, s_S^{(k)}$	$\in [0, 1]$ and $s_B^{(k)} + s_S^{(k)} \in [0, 1]$	Relative size of the pool in round k
$a_B^{(k)}, a_S^{(k)}$	$\in [0, 1]$	Attack level of the pool in round k

From round to round, the sizes of pools may change; and in fact it may happen that a bigger pool becomes a smaller pool in the next round. To be consistent with our terminology then, the salient feature we use to distinguish B from S is the assumption that $A_B \geq A_S$.

3.3 Choices

In each round, players simultaneously choose an attack level in $[0, 1]$. In round k , pool B chooses $a_B^{(k)}$, while pool S chooses $a_S^{(k)}$. The attack level is intended to be interpreted generically, independent of the specific attack form. The attack could be distributed denial of service (DDoS), or any other form of adversarial action that disrupts the attacked pool's mining efforts.

3.4 Consequences

The choices of mining pools affect both the short-term utilities of players, as well as the longer-term size of each pool as a result of miner migration.

Short-Term Consequences. Let $C \in [0, 1]$ be the per unit cost of an attack, then the utility of pool B in round k can be expressed in terms of the relative sizes of B and S via

$$u_B^{(k)} = \frac{s_B^{(k)} \cdot (1 - a_S^{(k)})}{1 - s_B^{(k)} \cdot a_S^{(k)} - s_S^{(k)} \cdot a_B^{(k)}} - C \cdot a_B^{(k)}. \quad (1)$$

In the above formula, $s_B^{(k)} \cdot (1 - a_S^{(k)})$ is the mining power of B considering S 's attack, $1 - s_B^{(k)} \cdot a_S^{(k)} - s_S^{(k)}$ is the mining power of the whole Bitcoin ecosystem considering both attacks, and $C \cdot a_B^{(k)}$ is the total cost of attack incurred by B .

The utility function is designed to correspond directly to the percentage of mining revenue obtain by the pool in the given round. The relative amount of coins being mined in round k is decreased by the two players' attacks, which explains the denominator; while the relative amount of coins being mined by

pool B in round k is affected proportionally to the attack level against pool B by pool S , which explains the numerator.

Note that by symmetry, we have the utility of S in round k as

$$u_S^{(k)} = \frac{s_S^{(k)} \cdot (1 - a_B^{(k)})}{1 - s_S^{(k)} \cdot a_B^{(k)} - s_B^{(k)} \cdot a_S^{(k)}} - C \cdot a_S^{(k)}. \quad (2)$$

Long-Term Consequences. Attack strategies also have long-term consequences, that do not affect the players' immediate revenue, but do affect miner migration, and hence the relative sizes of the pools in the next round.

In each round, the miners that are affected by an attack re-evaluate their choices and start to migrate. Formally, in each round, $s_B^{(k)} \cdot a_S^{(k)}$ (or $s_S^{(k)} \cdot a_B^{(k)}$) miners leave pool B (or S) due to attacks. The group of migrating miners redistribute themselves in the next round among the pools in a manner proportional to each pool's relative attractiveness level (A_B , A_S , or $1 - A_B - A_S$, for pool B , pool S , and all other pools, respectively).

Miners may also re-evaluate their choices from time to time even when they are not affected by an attack. Let $M \in [0, 1]$ denote the level of this base migration. If $M = 0$, then from any initial state $s_B^{(0)}, s_S^{(0)}$, the pool sizes remain constant from round to round when there are no attacks. If $M = 1$, then every miner re-evaluates her pool choice in each round. Similarly to migration due to attacks, the group of migrating miners redistribute themselves among the pools in a manner proportional to each pool's relative attractiveness level.

We may thus express the relative size of pool B in round $k + 1$ in terms of the relative sizes of pools in round k , together with attack levels and the base migration rate:

$$\begin{aligned} s_B^{(k+1)} &= s_B^{(k)} \\ &\quad + A_B \cdot [(1 - s_B^{(k)}) \cdot M + s_S^{(k)} a_B^{(k)} (1 - M)] \quad (\text{migration into } B) \\ &\quad - s_B^{(k)} \cdot (1 - A_B) \cdot [M + a_S^{(k)} (1 - M)] \quad (\text{migration out of } B). \end{aligned} \quad (3)$$

Analogously, the relative size of pool S in round $k + 1$ may be expressed as

$$\begin{aligned} s_S^{(k+1)} &= s_S^{(k)} \\ &\quad + A_S \cdot [(1 - s_S^{(k)}) \cdot M + s_B^{(k)} a_S^{(k)} (1 - M)] \quad (\text{migration into } S) \\ &\quad - s_S^{(k)} \cdot (1 - A_S) \cdot [M + a_B^{(k)} (1 - M)] \quad (\text{migration out of } S). \end{aligned} \quad (4)$$

4 Model Analysis

We begin the analysis of our modeling framework by proving a uniqueness result for each pool's relative size in a steady-state equilibrium.

4.1 Steady-State Pool Sizes

Theorem 1. If $M > 0$, then for any strategy profile (a_S, a_B) , there exists a unique pair of relative sizes (s_S^*, s_B^*) such that

$$(s_S^{(k+1)}, s_B^{(k+1)}) = (s_S^{(k)}, s_B^{(k)}) = (s_S^*, s_B^*).$$

Proof. Given a strategy profile (a_S, a_B) , the conditions of the theorem require (s_S^*, s_B^*) to satisfy

$$\begin{aligned} s_B^* \cdot (1 - A_B) \cdot [M + a_S(1 - M)] && \text{(migration out of } B\text{)} \\ = A_B \cdot [(1 - s_B^*) \cdot M + s_S^* a_B(1 - M)] && \text{(migration into } B\text{)} \end{aligned}$$

and

$$\begin{aligned} s_S^* \cdot (1 - A_S) \cdot [M + a_B(1 - M)] && \text{(migration out of } S\text{)} \\ = A_S \cdot [(1 - s_S^*) \cdot M + s_B^* a_S(1 - M)] && \text{(migration into } S\text{).} \end{aligned}$$

Solving B 's migration equation for s_B^* , we obtain

$$s_B^* = \frac{A_B \cdot [M + s_S^* a_B(1 - M)]}{M + a_S(1 - A_B)(1 - M)}.$$

This linear constraint is satisfied by all points (s_S, s_B) on the line segment connecting the points $\left(0, \frac{A_B M}{M + a_S(1 - A_B)(1 - M)}\right)$ and $\left(1, \frac{A_B[M + a_B(1 - M)]}{M + a_S(1 - A_B)(1 - M)}\right)$.

Similarly, the migration equation for S may be reduced to the linear constraint

$$s_S^* = \frac{-A_S M + s_B^* [M + a_B(1 - A_S)(1 - M)]}{A_S a_S(1 - M)},$$

which is satisfied by all pairs (s_S, s_B) on the line segment connecting the points $\left(0, \frac{-M}{a_S(1 - M)}\right)$ and $\left(1, \frac{(1 - A_S)[M + a_B(1 - M)]}{a_S A_S(1 - M)}\right)$.

We now wish to show that these two segments must intersect in the interval $[0, 1]$. More precisely, we claim that the second segment starts below the first segment when $s_S = 0$; and ends above the first segment when $s_S = 1$.

The two relevant inequalities are:

$$\frac{-M}{a_S(1 - M)} < \frac{A_B M}{M + a_S(1 - A_B)(1 - M)} \quad (5)$$

and

$$\frac{A_B[M + a_B(1 - M)]}{M + a_S(1 - A_B)(1 - M)} < \frac{(1 - A_S)[M + a_B(1 - M)]}{a_S A_S(1 - M)} \quad (6)$$

Inequality (5) follows because the first term is negative while the second term is positive (or zero if $A_B = 0$). Inequality (6) follows from:

$$\begin{aligned} \frac{A_B[M + a_B(1 - M)]}{M + a_S(1 - A_B)(1 - M)} &\leq \frac{(1 - A_S)[M + a_B(1 - M)]}{M + a_S A_S(1 - M)} \quad (A_S + A_B \leq 1) \\ &< \frac{(1 - A_S)[M + a_B(1 - M)]}{a_S A_S(1 - M)} \quad (M > 0). \end{aligned}$$

□

As a result of the theorem, the unique steady-state solution can be expressed directly in terms of the strategies a_S and a_B , the attractiveness levels A_S and A_B , and the migration constant M :

$$s_S^* = \frac{A_S M [M + a_S(1 - M)]}{[M + a_B(1 - A_S)(1 - M)][M + a_S(1 - A_B)(1 - M)] - A_B a_B A_S a_S (1 - M)^2} \quad (7)$$

$$s_B^* = \frac{A_B M [M + a_B(1 - M)]}{[M + a_B(1 - A_S)(1 - M)][M + a_S(1 - A_B)(1 - M)] - A_B a_B A_S a_S (1 - M)^2}. \quad (8)$$

Furthermore, we know that these values are in $[0, 1]$ under our modeling assumptions without having to do further case analysis.

4.2 Steady-State Pool Utilities

Since there is a unique pair of steady-state pool sizes for each strategy profile, we can find a steady-state equilibrium by assuming that the pool sizes and, hence, the players' utilities are given by Eqs. (7) and (8). In other words, given a strategy profile (a_S, a_B) , we may write the utility of each pool under the assumption that the relative sizes are the steady-state sizes s_S^* and s_B^* .

For pool S , we obtain

$$u_S = \frac{A_S M [M + a_S(1 - M)](1 - a_B)}{\text{Denominator}} - a_S C, \quad (9)$$

and for pool B , we have

$$u_B = \frac{A_B M [M + a_B(1 - M)](1 - a_S)}{\text{Denominator}} - a_B C, \quad (10)$$

where

$$\begin{aligned} \text{Denominator} = & M + a_S(1 - M - A_B)[M + a_B(1 - M - A_S)] \\ & + a_S a_B [(1 - A_S - A_B)(1 - M)^2 - A_S A_B]. \end{aligned} \quad (11)$$

This formulation will permit us to determine all the steady-state equilibria for the sequential game presented in Sect. 3 by finding Nash equilibria for a single-shot two-player game with the above utilities.

4.3 Peaceful Equilibria

We begin our analysis of equilibria by determining the conditions under which it is a stable strategy profile for each player to refrain from attacking the other player.

Theorem 2. *The strategy profile $(a_S, a_B) = (0, 0)$ is a Nash equilibrium just in case*

$$C \geq \frac{A_B A_S}{\min\{M, 1 - A_B, 1 - A_S\}}. \quad (12)$$

Proof. Suppose that $a_S = 0$. We want to characterize the conditions under which $a_B = 0$ is a best response. First, we express the utility of B in a steady state by substituting $a_S = 0$ into Eq. (10), obtaining

$$u_B = \frac{A_B[M + a_B(1 - M)]}{[M + a_B(1 - M - A_S)]} - a_B C. \quad (13)$$

When B does not attack ($a_B = 0$), her resulting steady-state utility is $u_B = A_B$; while if she attacks with full force ($a_B = 1$) her utility becomes $u_B = \frac{A_B}{1 - A_S} - C$. Because the utility function is analytic in a_B , any intermediate attack level can only be a utility-maximizing response strategy if the partial derivative of u_B with respect to a_B evaluated at that specific attack level is zero.

Computing the first and second partial derivatives of u_B with respect to a_B , we obtain

$$\frac{\partial u_B}{\partial a_B} = \frac{A_B A_S M}{[M + a_B(1 - M - A_S)]^2} - C, \quad (14)$$

and

$$\frac{\partial^2 u_B}{\partial a_B^2} = \frac{-2A_B A_S M (1 - M - A_S)}{[M + a_B(1 - M - A_S)]^3}. \quad (15)$$

Since the denominator of Eq. (15) is always positive, the second derivative itself is of constant sign; and this sign is negative if and only if $1 - M - A_S > 0$, or equivalently $M < 1 - A_S$. It is only in this case where the roots of the first derivative will give relevant maximizing solutions for the attack level.

In the case $M > 1 - A_S$, the roots of the first derivative will give minimizing attack levels, and in the case $M = 1 - A_S$ the first derivative will be constant, and hence the utility B will be maximized at either one of the endpoints of $[0, 1]$, or on the entire interval.

Setting the derivative from Eq. (14) equal to zero and solving for a_B , we obtain

$$a_B = \frac{\sqrt{\frac{A_B A_S M}{C}} - M}{1 - A_S - M}. \quad (16)$$

Now we consider two parameter cases.

- First, if $M \geq 1 - A_S$, then the maximizing attack level is one (or both) of the two endpoints 0 or 1. In this case an optimal response is $a_B = 0$ exactly when $A_B \geq \frac{A_B}{1-A_S} - C$, or equivalently, when

$$C \geq \frac{A_B A_S}{1 - A_S} = \frac{A_B A_S}{\min\{M, 1 - A_S\}}.$$

- Second, if $M < 1 - A_S$, then the maximizing attack level will be zero if and only if the point at which the derivative is zero is non-positive. Since we are in the case $1 - A_S < M$, we may deduce from Eq. (16) that the analytically-maximizing a_B is non-positive if and only if $\sqrt{\frac{A_B A_S M}{C}} - M \geq 0$. This condition reduces to

$$C \geq \frac{A_B A_S}{M} = \frac{A_B A_S}{\min\{M, 1 - A_S\}}.$$

These two parameter cases exhaust all options; and we have shown that in each case $a_B = 0$ is a best response to $a_S = 0$ if and only if

$$C \geq \frac{A_B A_S}{\min\{M, 1 - A_S\}}.$$

To have $(a_S, a_B) = (0, 0)$ be an equilibrium, we also need $a_S = 0$ to be a best response to $a_B = 0$. By symmetry, this will happen if and only if

$$C \geq \frac{A_B A_S}{\min\{M, 1 - A_B\}}.$$

We conclude that a peaceful equilibrium exists if and only if

$$C \geq \frac{A_B A_S}{\min\{M, 1 - A_B, 1 - A_S\}}.$$

□

4.4 One-Sided Attack Equilibria

Our next special case to consider is when exactly one of the players attacks while the other remains peaceful.

Theorem 3. *The strategy profile $(a_S, a_B) = (0, 1)$ forms a Nash equilibrium if and only if*

$$C \leq \frac{A_B A_S}{(1 - A_S)^2} \cdot \min\{M, 1 - A_S\} \tag{17}$$

Proof. First suppose that $a_B = 1$. Then the utility of S is given by

$$u_S = -a_S C, \tag{18}$$

and this quantity is clearly maximized for $a_S \in [0, 1]$ by taking $a_S = 0$. So $a_S = 0$ is always a best response to $a_B = 1$.

Next suppose that $a_S = 0$. We want to characterize the conditions under which $a_B = 1$ is a best response. From the previous special case analysis, we have

$$u_B = \frac{A_B[M + a_B(1 - M)]}{[M + a_B(1 - M - A_S)]} - a_B C.$$

Exactly as in the previous analysis, when $a_B = 0$, we have $u_B = A_B$; and if $a_B = 1$, we get $u_B = \frac{A_B}{1 - A_S} - C$. The same conditions applied to the derivative of u_B determine when the maximizing value of a_B is reached at a boundary point (0 or 1) or whether it may relate to where the derivative is zero at

$$a_B = \frac{\sqrt{\frac{A_B A_S M}{C}} - M}{1 - A_S - M}.$$

The derivative value is relevant if and only if $M < 1 - A_S$.

In the case where $M \geq 1 - A_S$, a maximizing attack level is one of the endpoints of $[0,1]$; and in this parameter case, the best response is $a_B = 1$ if and only if

$$C \leq \frac{A_B A_S}{1 - A_S} = \frac{A_B A_S}{(1 - A_S)^2} \cdot \min\{M, 1 - A_S\}.$$

In the case $M < 1 - A_S$, the maximizing value is 1 if and only if the global analytically-derived maximum is at least 1. This happens when

$$\begin{aligned} \frac{\sqrt{\frac{A_B A_S M}{C}} - M}{1 - A_S - M} &\geq 1 \\ \sqrt{\frac{A_B A_S M}{C}} - M &\geq 1 - A_S - M \\ \frac{A_B A_S M}{C} &\geq (1 - A_S)^2 \\ \frac{A_B A_S M}{(1 - A_S)^2} &\geq C \\ C &\leq \frac{A_B A_S}{(1 - A_S)^2} \cdot \min\{M, 1 - A_S\} \end{aligned}$$

Again these two parameter cases exhaust all options; and in each case $(a_S, a_B) = (0, 1)$ is an equilibrium configuration if and only if

$$C \leq \frac{A_B A_S}{(1 - A_S)^2} \cdot \min\{M, 1 - A_S\}.$$

□

To have $(a_S, a_B) = (1, 0)$ be an equilibrium, we would need the condition

$$C \leq \frac{A_B A_S}{(1 - A_B)^2} \cdot \min\{M, 1 - A_B\}.$$

Of course both conditions may be simultaneously satisfied for C sufficiently small, in which case both one-sided attack configurations will be equilibria.

5 Numerical Illustrations

5.1 The Peaceful Equilibrium

Figure 1 shows the parameter combinations where the peaceful equilibrium exists (i.e., $a_S = 0$ and $a_B = 0$ being best responses to each other). In Fig. 1(a), we fix the parameters $A_S = 0.2$ and $A_B = 0.3$, and we vary the parameters M and C . The figure shows that, if both M and C are high, then the peaceful equilibrium is possible; however, if either of these parameters is low, then there can be no peace. The latter is especially important in the case of M , which has no effect on the existence of the peaceful equilibrium once its value reaches $1 - A_j$ (0.7 in the figure). In practice, this means that both the pools and the users have to act in order to reach the peaceful equilibrium: the pools have to employ defensive countermeasures which increase C , while the users have to be proactive in their mining-pool choice, which increases M .

In Fig. 1(b), we fix the parameters $M = 0.5$ and $C = 0.1$, and we vary the parameters A_S and A_B . The figure shows that the peaceful equilibrium exists if either one (or both) of the pools has a low attractiveness. In practice, this

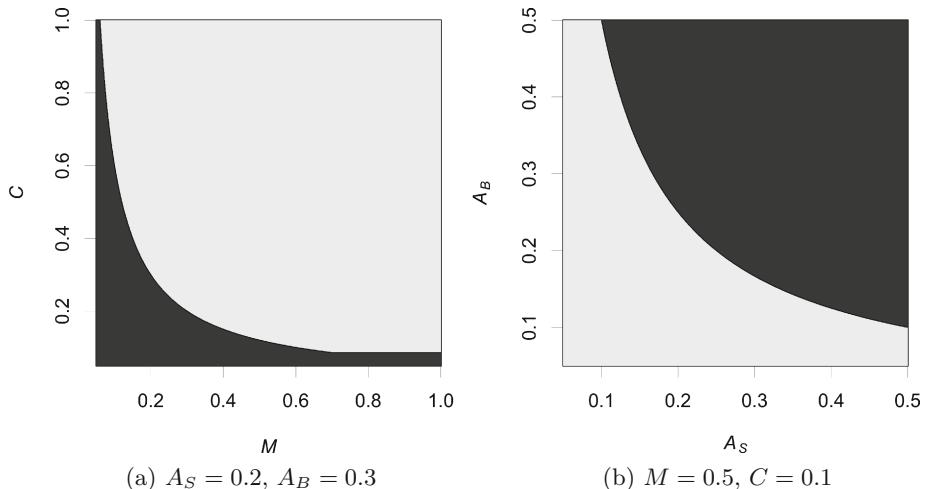


Fig. 1. Existence of the peaceful equilibrium (i.e., $a_S = 0$ and $a_B = 0$). Lighter shaded areas represent parameter combinations where the peaceful equilibrium exists.

means that a healthy competition between the pools, in which both of them try to attract miners, is very beneficial: not only will it result in better deals for the miners, but it may also bring peace.

5.2 One-Sided Attack Equilibria

Figure 2 shows the parameter combinations where one-sided attack equilibria exist (i.e., when pool $i \in \{S, B\}$ playing $a_i = 1$ and the other pool playing $a_{\bar{i}} = 0$ forms an equilibrium). In Fig. 2, we fix the parameters $A_S = 0.2$ and $A_B = 0.3$, and we vary the parameters M and C . The figure shows that one-sided attack equilibria are more likely to exist when M is high but C is low. This means that, to avoid a one-sided attack equilibrium, the pools must employ defensive countermeasures that increase the cost of attack C . Furthermore, the figure also shows that less attractive pools are more likely to play a one-sided attack strategy (darker shaded middle section representing $(a_S = 1, a_B = 0)$). While this may seem counterintuitive at first, it is actually very easy to explain. The more attractive pool has more miners even without launching an attack; hence, it is less inclined to dominate the other player with a marginalizing attack. The less attractive pool, on the other hand, has a lot to gain from such an attack; hence, it is more inclined to try to “steal” the miners of the more attractive pool.

In Fig. 2(b), we fix the parameters $M = 0.5$ and $C = 0.1$, and we vary the parameters A_S and A_B . The figure shows that, once a pool is highly attractive to the miners, the other pool will have an incentive to launch a marginalizing attack against it. While attacks are generally harmful to the Bitcoin ecosystem, they have positive effects in this context, as they prevent one pool from growing too large.

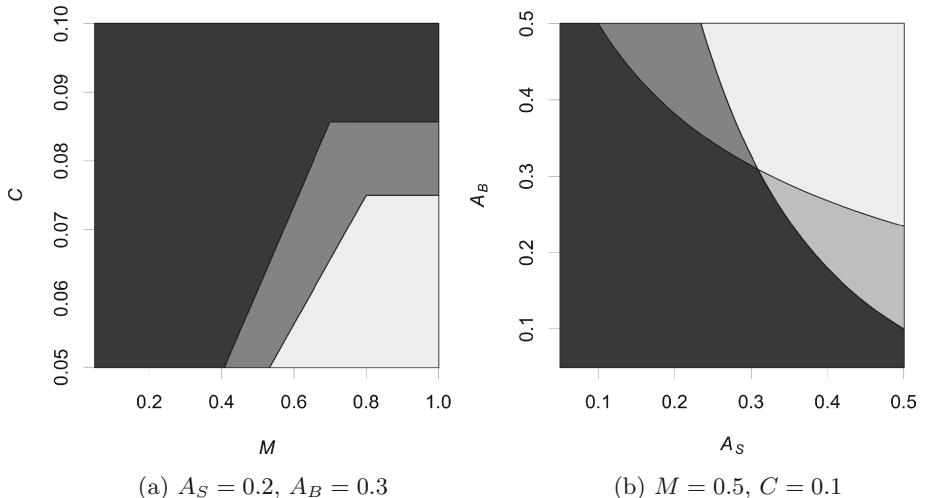


Fig. 2. Existence of one-sided attack equilibria. Shades from darker to lighter: no one-sided attack equilibrium, $(a_S = 1, a_B = 0)$ forms an equilibrium, $(a_S = 0, a_B = 1)$ forms an equilibrium, both form equilibria.

6 Conclusion and Future Work

In this paper, we proposed a game-theoretic model of attacks between Bitcoin mining pools, which – to the best of our knowledge – is the first study to consider long-term consequences. The analysis of our model has revealed a number of interesting implications for making the Bitcoin ecosystem more viable. We have seen that, in order to make the peaceful equilibrium viable, the unit cost of attack and the miners' base migration rate both have to be increased, and no pool can have an overwhelming attractiveness. We have seen that these factors also help preventing a marginalizing attack against one pool.

We limited the mining pools' strategic choices to launching attacks and assumed that the effects of defensive countermeasures are incorporated into the unit cost of attack. In future work, we can extend this model by allowing the pools to deploy additional defenses for some fixed cost, which decrease the effectiveness of attacks. As another direction, we can also extend the model by allowing the pools to choose some of the parameters that affect their attractiveness levels. For example, a pool could choose to increase its fee, which decreases its attractiveness but increases its utility for a given steady-state size. Finally, in this paper, we modeled only two pools as strategic players, but we intend to extend our work towards the case of multiple mining pools.

Acknowledgments. We thank the reviewers for their detailed feedback. This work was supported in part by the National Science Foundation under Award CNS-1238959.

References

1. Bensoussan, A., Kantarcio glu, M., Hoe, S.R.C.: A game-theoretical approach for finding optimal strategies in a botnet defense model. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec 2010. LNCS, vol. 6442, pp. 135–148. Springer, Heidelberg (2010)
2. Böhme, R., Christin, N., Edelman, B., Moore, T.: Bitcoin. *J. Econ. Perspect.* (forthcoming)
3. Christin, N., Grossklags, J., Chuang, J.: Near rationality and competitive equilibria in networked systems. In: Proceedings of the ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems, pp. 213–219 (2004)
4. Clark, D., Konrad, K.: Asymmetric conflict: weakest link against best shot. *J. Conflict Resolut.* **51**(3), 457–469 (2007)
5. Fultz, N., Grossklags, J.: Blue versus red: towards a model of distributed security attacks. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 167–183. Springer, Heidelberg (2009)
6. Goldfarb, A.: The medium-term effects of unavailability. *Quant. Mark. Econ.* **4**(2), 143–171 (2006)
7. Grossklags, J., Christin, N., Chuang, J.: Secure or insure? A game-theoretic analysis of information security games. In: Proceedings of the 2008 World Wide Web Conference (WWW 2008), pp. 209–218, April 2008

8. Huang, D.Y., Dharmdasani, H., Meiklejohn, S., Dave, V., Grier, C., McCoy, D., Savage, S., Weaver, N., Snoeren, A.C., Levchenko, K.: Botcoin: monetizing stolen cycles. In: Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS) (2014)
9. Johnson, B., Laszka, A., Grossklags, J., Vasek, M., Moore, T.: Game-theoretic analysis of DDoS attacks against bitcoin mining pools. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 72–86. Springer, Heidelberg (2014)
10. Laszka, A., Felegyhazi, M., Buttyán, L.: A survey of interdependent information security games. ACM Comput. Surv. **47**(2), 23:1–23:38 (2014)
11. Li, Z., Liao, Q., Blaich, A., Striegel, A.: Fighting botnets with economic uncertainty. Secur. Commun. Networks **4**(10), 1104–1113 (2011)
12. Litke, P., Stewart, J.: BGP hijacking for cryptocurrency profit, August 2014. <http://www.secureworks.com/cyber-threat-intelligence/threats/bgp-hijacking-for-cryptocurrency-profit/>
13. Liu, P., Zang, W., Yu, M.: Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM Trans. Inf. Syst. Secur. **8**(1), 78–118 (2005)
14. Manshaei, M., Zhu, Q., Alpcan, T., Bacşar, T., Hubaux, J.P.: Game theory meets network security and privacy. ACM Comput. Surv. **45**(3), 25:1–25:39 (2013)
15. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008). <http://bitcoin.org/bitcoin.pdf>
16. Plohmann, D., Gerhards-Padilla, E.: Case study of the miner botnet. In: Proceedings of the 4th International Conference on Cyber Conflict (CYCON), pp. 345–360 (2012)
17. Schechter, S.E., Smith, M.D.: How much security is enough to stop a thief? In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 122–137. Springer, Heidelberg (2003)
18. Schelling, T.: The Strategy of Conflict. Oxford University Press, Oxford (1965)
19. Spyridopoulos, T., Karanikas, G., Tryfonas, T., Oikonomou, G.: A game theoretic defence framework against DoS/DDoS cyber attacks. Comput. Secur. **38**, 39–50 (2013)
20. Vasek, M., Thornton, M., Moore, T.: Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 57–71. Springer, Heidelberg (2014)
21. Wu, Q., Shiva, S., Roy, S., Ellis, C., Datla, V.: On modeling and simulation of game theory-based defense mechanisms against DoS and DDoS attacks. In: Proceedings of the 2010 Spring Simulation Multiconference, pp. 159:1–159:8 (2010)

Issues in Designing a Bitcoin-like Community Currency

David Vandervort^(✉), Dale Gaucas, and Robert St Jacques

PARC, Webster, NY, USA

{david.vandervort, dale.gaucas,
robert.stjacques}@xerox.com

Abstract. The invention of the Bitcoin protocol has opened the door to new forms of financial interaction. One such form may be to adapt Bitcoin technology for use as a community currency. A community currency is a form of money issued by a non-government entity to serve the economic or social interests of a group of people, often in a small geographic area. We propose a model of a community cryptocurrency that includes a community fund from which community members may take out loans if the community votes to approve them. We consider possible vulnerabilities and mitigations to issues that would affect this community fund, including issues of identity, voting protocols and funds management. We conclude that these vulnerabilities are, in most cases, amenable to technological mitigations that must be adaptable to both community values and changing conditions, emphasizing the need for careful currency design.

1 Background

Money issued by national governments is a dominant fixture of modern economic exchange. So called fiat currency is freely traded for goods and services, with its printing, issue and acceptance protected by law. Bitcoin and its derivatives are a different kind of currency, useable for many purchases despite being non-government issued and protected by no law. Community currencies, including such subtypes as Local Exchange Trading Systems (LETS), time banks and business trade exchanges, are similar non-state issued moneys that circulate in parallel with fiat currencies [1]. Purposes of community currencies often go beyond economic exchange to supporting values and causes including social, environmental or ethical dimensions [2].

Examples of community currencies include the Brixton Pound, Ithaca Hours and BerkShares. Each serves a different local area and evinces a different philosophy of society and economics in its construction. Bitcoin is a digital currency that was developed in order to remove the need for a central authority (i.e. banks) to mediate and clear transactions and to protect the privacy of those engaging in transactions [3]. This is clearly also a philosophy. In a sense, Bitcoin can be regarded as a community currency, with the community being those people who care about Bitcoin. They have shown the ability to work together toward goals large and small, to hold informative discussions, to build an economy and to help each other in times of need. It seems reasonable then, to consider ways in which Bitcoin technology could benefit other communities.

1.1 Cryptocurrencies

Because of its use of cryptographic methods to protect the integrity of transactions and of the currency itself, Bitcoin is known as a cryptocurrency. Advantages of Bitcoin include immutability of transactions, pseudonymity, distributed control that prevents manipulation by a central authority, complete transparency (as anyone, anywhere can download the blockchain and view all transactions) and strong cryptographic protection against tampering.

While Bitcoin was the first cryptocurrency, it was quickly followed by an explosion of currencies based on the same or very similar technology. There are a plethora of cryptocoins available besides Bitcoin, including Litecoin, Mastercoin, Primecoin, Marscoin, Zerocoin, Dogecoin, Reddcoin and many others. Many of these experiment with slight differences in the protocol to serve perceived needs of the community and the world.

Bitcoin is a software based system. Bitcoin transactions manipulate data to exchange ownership of bitcoins between addresses, with no requirement for physical exchanges of notes or coins. This allows complex scripting of behaviors, such as m of n (multi-signature) transactions and smart contracts [4]. It also means that any behavior that can be expressed in code can theoretically be encoded into a Bitcoin-like protocol. If this is true, then features found in community currencies may be added to Bitcoin to make something new.

2 Community Cryptocurrency Features

Community currencies include features beyond direct economic value that are intended to advance their goals. Two important features for the present discussion are demurrage and the maintenance of a fund for loans or grants. Demurrage is the practice of reducing the value of currency in proportion to the time it is held, rather than spent. Reportedly, the “peanuts” LETS currency in Chiba Prefecture in Japan charges a 1% fee per month on currency that is not used. Demurrage encourages people to keep circulating currency so as to avoid the loss of value. This is reportedly a significant factor in the success of Peanuts [5]. Note that the velocity of a community currency (roughly, the number of times a single note or coin is re-spent in the economy) can be quite high and demurrage is sometimes cited as one of the reasons [6].

A loan or grant fund is possibly one of the most powerful tools of development possessed by community currencies. The BerkShares currency maintains a loan fund for local businesses [7]. It is not unusual for hours based systems such as Ithaca Hours and Calgary dollars also to provide small loans or outright grants to local businesses [8]. Small business loans can be a driver of economic development. As well, personal loans and grants can be tools for assisting those in need. Loans can also be targeted at types of businesses, or interest rates tailored to meet social as well as economic goals.

Other possible features for community currencies include restriction to a small geographic area (geofencing), privileged transactions, interest payments and participant dividends. Because the last two features must have a source of funds, their implementation will likely involve draws from the same fund as loans and grants. For that

reason they will not be dealt with at length. The focus here is on the community loan and grant fund. For simplicity, this fund will be referred to in most instances as the community fund.

There have been several forays into using Bitcoin technology for community currencies. Examples include the following.

- Mazacoin (<http://mazacoin.org>). Mazacoin claims to be the national currency of the Lakota Nation, though it is unclear from reports if the officials of the nation share this view. The creator of Mazacoin pre-mined 25 million coins (meaning he created them before allowing others to mine for their own) to be set aside for a tribal fund that would give grants to individuals, businesses and non-profits focused on the tribe [9].
- IrishCoin (<http://IrishCoin.org/>). IrishCoin is targeted specifically at promoting tourism to Ireland and has allocated 7% of the total volume of the coin for distribution to businesses and organizations associated with that industry for use as a “discount token” [10].
- Marscoin (<http://marscoin.org>). This coin has the unusual goal of becoming the currency used by colonists on Mars.¹ Four hundred thousand coins were pre-mined and donated to the Mars Society, a not-for-profit organization that seeks to establish a colony on Mars. The eventual goal is for colonists to take the Marscoin blockchain with them to Mars and use it as the basis for a local economy [11].

These and similar examples of community cryptocurrencies adapt the Bitcoin protocol to serve their needs without significant new features. More extensive adaptations of existing Bitcoin features and new capabilities can increase differentiation and utility for community cryptocurrencies. The remainder of this paper will discuss integrating these extended features into a cryptocurrency so it may serve an individual community. A significant portion will be devoted to a vulnerability analysis of the community fund and to methods of community decision making, centered around the community fund.

2.1 Mining

One of the protections Bitcoin has against fraud and manipulation is that coins are created and transactions verified in a distributed manner. All the working nodes check each other’s work. It is, however, a consensus algorithm, with the blockchain reflecting work that the majority of nodes agree on. This makes it vulnerable to what is known as a 51% attack. In this attack, one person, node or mining pool acquires enough power (possibly through having more or better hardware than other nodes) to force a consensus on its own terms. Thus this powerful unit can conceivably corner the market on coin creation or even insert fraudulent transactions into the blockchain [12].

In a community cryptocurrency limits to the participant pool imposed by geography, interest, or other factors may increase this risk. Careful attention must therefore be paid to the numbers of mining and verification nodes. It may be then that proof-of-stake algorithms may be safer for community cryptocurrencies than the Bitcoin

¹ One of the authors of this paper (Vandervort) has mined Marscoin. His wallet currently holds 321.824521 Marscoin. He has no plan to go to Mars.

proof-of-work method. Proof-of-stake protocols require participants to prove possession of some amount of the currency for a minimum period of time before being permitted to produce new blocks (and with them, new coins) [13]. One way of jump-starting this is for a small amount of currency to be automatically given to new members of the community, probably from the community fund. Membership may be determined simply by downloading a new wallet, registering a new identity (discussed below) or some other method.

2.2 Geofencing

Community currencies are often intended to serve a local geographic area. BerkShares and Ithaca Hours are examples of these kinds of community currencies. Implementing geographic limitations in a cryptocurrency may have wide ranging consequences and difficulties.

Thanks to the revolution in geopositioning systems (GPS), software can be aware of the location where it is being used. This is not universal as location is often considered private data and many people block it by default. For a community cryptocurrency, location data can be used to verify the location of transactions and even software downloads. However, locations can be spoofed, for example by accessing a download site through a virtual private network. The question of how to handle offline transactions, which may experience delays before being committed to the blockchain is also an issue, since verification of location information may not necessarily occur at the time of the transaction.

Even putting aside the possibility of spoofing IP addresses and other location identifiers, there are issues with geofencing a digital currency. Enforcing the restrictions means forcing both businesses that accept the currency and users who spend it to reveal location information. Many may find this intrusive and the pool of available users will then be reduced accordingly. The size of the area and the mobility of people within it is also an issue. What happens to someone who travels outside the area briefly then realizes a bill needs to be paid? Is the payment prevented from going through until the payer returns home? There are many other cases that could be imagined in which geographic restrictions are an impediment even to people who live and work within the assigned area. Softer restrictions that allow transactions outside the defined area seem more supportable but risk allowing the area to artificially widen. This may not be a disadvantage in practice as it allows the pool of participants to widen as well.

Mining for new coins is a different question. Should this be allowed outside the intended area? If communities answer yes, they run the risk that outsiders will come to dominate mining, removing control of the currency from its intended community. If, however, they discourage this option, the total number of mining nodes may be too low to keep the currency stable or to fend off attempts to take over 51% of the processing power.

It can be seen then, that enforcing geographic limits at the protocol or software levels may create complications for a currency and its users. This indicates that the best course may be for real human beings to concentrate on working with their neighbors and with local businesses to make their currency popular in the local region rather than to use technology to enforce geographic restrictions. Therefore, at the current time geofencing related features are not recommended for community cryptocurrencies.

2.3 Privileged Transactions

Privileged transactions are those that the community encourages by providing extra incentives. These transactions are considered to advance community goals or express community values. Examples include giving bonus payments for services performed for the elderly, or discounts for purchases of environmentally friendly products. In each case, for the economic equation to work, the difference between the normal price and the privileged price must be made up from somewhere. The most logical source for these additional funds is the community fund (discussed below).

As a direct expression of the community's values, privileged transactions are a means of fostering community cohesion. Including this feature in a digital currency requires some method of indicating what types of transactions would be privileged and how much privilege they would receive. Privileges expressed numerically, such as discounts and bonuses, are the simplest to translate into rules that can be interpreted by software. Less deterministic privileges, such as a promise of invitations to dinner at some time in the future, might be specified by text strings but automating verification of their delivery is difficult. The Bitcoin protocol may enable creative solutions to this problem. For example, a promise of dinner can be encoded as a very small multi-signature transaction, that is completed when all parties are satisfied that the promise has been kept. Verification of some sort is important for the sake of transparency. When users can check in the blockchain to see that promises are being kept, their faith in the currency and the community is likely to be greater than cases where there is no such verification.

The problem of verification brings up another issue that is important to the design and function of a community currency: trust. In some communities, methods of verification might be relaxed as a show of trust among community members. In such communities it might be enough for someone to send an email to one of the community leaders describing the privileged transactions they have been involved in and asking for bonuses thus earned. Particularly in small groups where the members have considerable face-to-face contact this kind of informality may be acceptable. Whether this type of small, trusting community needs a cryptocurrency is another question. In any case, the ability to automatically adjust compensation for different types of transactions and to verify the accuracy and nature of payments is a significant advantage of software-based systems over more traditional paper currencies or even many electronic exchanges. The convenience of having an account automatically credited by the correct amount the moment the transaction takes place, rather than having to go to a local "bank" and exchange notes or access a website and enter verification details is a significant advantage of a Bitcoin-based model for these currencies.

2.4 Demurrage

In order to encourage economic transactions, some currencies use demurrage, meaning they reduce the value of unspent notes over time. This gives people holding them incentive to move them quickly in order to capture as much value as possible. This in turn may magnify the economic multiplier effect (or velocity) of such currencies.

Though many factors may affect the velocity of a currency, demurrage appears to have been at least somewhat effective for several community currencies [14] making it a potentially desirable feature.

Administering demurrage means that the time of transfer of each note must be recorded so that the value can be properly calculated. In the physical world, this means that either a note must have a timestamp (or series of timestamps) on its face, or it must have an identifier such as a serial number that can be associated with the timestamps in a central registry. This second method of tracking time for notes and transactions is similar to the function of the Bitcoin blockchain, which directly incorporates timestamps into transaction block data [3]. There are, however, potential pitfalls. Differences in time zones, system clocks and even the representation of time in different programming languages may make it impractical to calculate reductions in value over short periods of time. Recalculating the value of a particular coin should probably be done on a scale of days or weeks rather than seconds or minutes.

It is essential, also, that changes in the value of currency be verified at the mining level, similar to the way transactions are incorporated into the blockchain. In fact, the simplest implementation is to remove some portion of currency at the time it is used and deposit that portion into the community fund. This implementation prevents unintentional destruction of the total value of the currency, which could adversely affect the stability of the currency over time. It could also help to keep the balance of the community fund healthy.

The question arises of the relationship between demurrage and a proof-of-stake system. One of the advantages of proof-of-stake is that it may allow anyone who has a stake to mine new coins. Typically, coins must be shown not just to exist but to be reasonably “fresh” [13]. If the rate at which demurrage removes value is too fast, it could then interfere with the ability to show stake, both by directly removing coins that would otherwise show stake and by encouraging people to spend their coins so quickly that they have little or no stake in their wallets for verification. Yet, if demurrage is too slow, it provides little incentive for people to spend their coins, defeating its purpose. Thus if both proof-of-stake and demurrage are implemented in the same currency, the rate of change must be carefully calibrated to encourage spending while preserving stake²

Related to demurrage is the payment of interest, which increases holdings over time rather than decreasing them. The money to pay interest must come from some source. That source is most probably the community fund. Interest incentivizes saving rather than spending, which may not be in the best interest of the community economy. However, it also provides clear value, which may improve trust in the community. It can be used to demonstrate proof-of-stake by adding “fresh” currency to a wallet. Variation in interest rates, such as a reduction in interest payments when the community fund has a low balance, may offset the benefits, however.

² At the time of this writing (September 2014), a patent application titled “Peer-to-peer (p2p) currency platform implementing demurrage,” (USPTO patent application number 20130346164) may affect implementations of demurrage. The application can be viewed at <http://appft1.uspto.gov/netacgi/nph-Parser?Section1=PTO1&Sect.2=H1TOFF&d=PG01&p=1&u=/netahmt/PTO/srchnum.html&r=1&t=G&l=50&s1=20130346164 PGNR>

2.5 The Community Loan Fund

Some community currencies maintain a fund that can be used to make small grants or loans, usually for the purposes of starting or improving small businesses. Such funds can be effective at building owner-operated businesses [15], therefore incorporation of loans or grants into a community currency may contribute toward community goals. There are two questions to consider in designing this capability. Where do the funds come from, and how are decisions made concerning their disbursement? We propose that the answers to both questions be incorporated directly into the currency software.

2.5.1 Adding to the Community Fund

In cryptocurrencies, the most common method of stocking the community fund is currently for the creator(s) to pre-mine some amount that they can keep under their control. This method is easy to implement by simply running the first mining node or a small number of such nodes, without allowing others to download the software and run their own nodes, until a sum deemed sufficient has been mined. The cryptocurrency community in general tends to frown on this practice since it allows an unscrupulous operator to introduce a currency that they control from the very beginning. Note again the issue of trust comes into play.

It is possible for cryptocurrencies to add coins to a general fund in other ways, for example by adding a small fee to all transactions, or to all transactions above a certain amount, which will be paid to an address associated with the fund. In a corollary to privileged transactions, it may be possible to charge an extra fee for discouraged transactions, such as buying gasoline, if the goals of the community are environmentally oriented. The Bitcoin protocol already uses transaction fees as a means of compensating miners. Adding another fee or increasing the fee slightly and splitting between two recipients are relatively simple modifications that can support a community fund without pre-mining.

A related method of adding currency to the community fund is to take a small portion of mining rewards for the fund. In current cryptocurrencies, a node that is the first to generate a solution for a block is given a reward in new coins. This is called mining the currency. In the Bitcoin network the current reward is 25 bitcoins per block. In a community cryptocurrency, splitting off a portion of the block reward for the community fund is feasible. Miner objections may be reduced if the amount remaining to them is enough for a profit, or if they perceive some other benefit such as a good reputation within the community. The reputation factor could be enhanced by allowing miners to adjust the amount deposited to the community fund, therefore making it more a donation than an involuntary side effect. Designers of community cryptocurrencies may find it advisable to set a minimum donation, rather than depending entirely on the altruism of the miners.

2.5.2 Disbursing from the Community Fund

In traditional finance, the most common method of disbursing funds for loans and grants has been for a small number of administrative persons to make all decisions. Even in community currencies, this seems to be the default approach. So, for example, the creators of IrishCoin stocked a distribution fund by pre-mining and indicated a

preference for distributing it to businesses and organizations associated with Irish tourism [10]. This approach requires no custom programming to implement in the current Bitcoin protocol.

Another method can be built that uses the distributed nature of the protocol to take the disbursement out of the hands of a few members and give it to the whole community. This would involve a multi-step process. First, a transaction of a new type, community loan, is created by any authorized user, which in many communities may include any member of the community. The amount of the transaction is the amount of the loan (or grant) from the community fund. Then members of the community submit transactions of another new type, vote. Each vote either approves or disapproves of the transaction. When a threshold is reached, the vote is finalized. If the loan is approved the funds are released to the address specified. If the loan is disapproved, the transaction is invalidated. The voting threshold for or against the loan will vary from community to community. Some will require a majority of voting members. Others will require a supermajority. Any amount that can be mathematically described can be conceived.

The advantages of a system where the community votes on the disbursement of funds are in increased trust among community members and commensurately increased investment in the goals and activities of the community. There are numerous difficulties created by the proposed system as well. The next sections of this paper will discuss the problem of identifying “voting” members of the community as well as the recipient of proposed loans. This will be followed by a discussion of potential vulnerabilities to the integrity of the community fund.

3 Challenges with a Cryptocurrency Community Fund

The community fund and votes concerning loans from it are where the community works together, expressing shared values and building economic and social structures to make the community stronger. Conversely, should the loan fund become depleted or weaknesses in the system of proposing and voting on loans develop, the community could suffer a loss of trust, cohesion and even economic viability. Our analysis identified three major areas where design must be carefully considered in order for a community fund regulated by community participation to be viable. Those areas are identification of community members, tallying of votes and regulation of loans. The issues related to these factors are often interrelated.

3.1 Identity

The original Bitcoin system is highly successful at allowing relatively anonymous, trustless transactions. In a community currency there are at least three reasons why identity might be revealed to some extent. It may be necessary to ensure the proper counting of votes, to validate the recipient of grants and loans and to find businesses that will accept the currency in payment. Serving these purposes may require different levels of disclosure of identifying information. For example, while identifying a loan recipient may require a full name and address, voting may require no more than a unique identifier. In other cases, some third party identifier or certificate issued by a provider

who possesses but does not share more specific identification information, may be a good compromise between the extremes of full identification and full anonymity.

The level and type of identifier used may also vary depending on purposes. Businesses may prefer to be more open about their true name and location than people whose purpose is not business related. Registering clear identifying information as a public identity is a simple form of advertising. In communities with a strong local component, geographic coordinates might also be part of an identifier.

How identity data is stored and accessed is an important consideration. Do identities need to be registered directly in the blockchain (or some blockchain) or is it enough to have some identifier such as a username associated with a wallet address? Could a separate blockchain for identity be used, with transaction meta-data incorporating a hash of a location in the identity chain? In this case, is it enough to reference identities registered with some service such as Namecoin? While this low level of identification is suitable for many purposes, it seems likely that a higher level of disclosure is required in communities that vote on local issues, or in which reputation is important.

3.2 Voting

Voting is integral to the model of community currency being developed here. Voting is one way that people participate in the community which ties it closely to identity. In order to verify that only people who belong to the community cast votes, voters must be identified in some way. If, however, the community requires votes to be cast in secret, verification and tallying become separate steps. When using a structure such as a blockchain to store votes, meeting the goals of both secrecy and identity verification is a difficult technical problem.

In the physical world, voting is usually restricted to a specific time period. This makes sense especially when dealing with monetary loans, which may be time sensitive. Rules for determining winning and losing vary with locality and context. For a political election, the highest number of votes may win, or a majority ($>50\%$) may be required with lower numbers resulting in a runoff. In a jury, unanimity may be required. In a legislature, some actions may pass by majority vote while others may require a supermajority. An additional concept to consider is the need for a quorum. This is the smallest number or percentage of the membership that must vote for results to be considered valid. Without a reasonable number for a quorum, a community risks having its resources come under the control of a small number of members. All of these properties can be modeled in the community cryptocurrency software. As discussed above, votes are treated in this system as another form of transaction. Therefore, mining nodes may apply rules to voting transactions and voting blocks as the community requires, though choosing optimal values may be challenging.

3.3 Loan Regulation

In order for a loan fund to be viable, it must not loan out more currency than it has coming in and loans must be paid back in a timely manner. The way loans are proposed to and approved by the community influences these factors. Communities should

consider factors such as who is allowed to propose loans. Should anyone be allowed or should it be restricted to members in good standing? Does a proposed loan have to be seconded before it is put up for a vote to the whole community? Should there be a maximum size to loans either in whole numbers or a percentage of available funds? Should loan recipients be allowed to receive new loans while a previous loan has not yet been paid back?

There must be some method of stocking the community loan fund. As mentioned above, methods include taking tithes from transactions or from miners. There may also be a tax on wallet balances above a certain amount or membership fees for businesses. Whatever method is used to fill the fund, it must be carefully balanced with outgoing loans. If the community fund becomes too large it may stifle other economic activity effectively creating deflation. If it is too small, loans will be choked off and economic growth may suffer.

In addition to the need to encode policy into software comes the question of how to change policy. Any policy may need to be revised to accommodate real world experiences. Perhaps if too many loans default, a temporary cap needs to be put in place, or certain members need to be permanently banned from proposing or receiving loans or grants. Creating a system that is comprehensive, secure and also responsive is a significant challenge for both currency designers and software developers.

4 Vulnerability Assessment

While Bitcoin transactions provide a level of protection of user identity, in a community fund this may be compromised to some extent. Businesses, including those operated by the self-employed, may desire more visibility to the community in order to advertise their services and so their names may accrue a good reputation. The counting of votes is tied to identity as well. Vote tallies must be accurate and, in many communities, secret, in order to ensure that the will of the community as to the disbursement of funds is not subverted. Similarly, the regulation of loans refers to the way in which loans are proposed, disbursed and even repaid.

In a community in which all the members know and trust each other, these problems may be ignored. In more common communities where self-interest sometimes conflicts with the community interest, controls are needed to mitigate potential risks. We analyze the risks from the point of view of the STRIDE framework, slightly modified to accommodate analysis at the fund operation level rather than at the point of software implementation.

4.1 STRIDE Framework

STRIDE is an acronym for a common set of risks to software-based systems. It stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of Privilege [16]. Our analysis, however, is not of the software implementation but of potential risks at the community fund level, as the structure of the system must be determined before being implemented in software. Therefore we use the terms,

plus one other, for structural issues that may impact fund operations. This gives the following categories for the analysis.

- Spoofing. When a person pretends to be another, the pretend identity is said to be spoofed. In this case, one person may spoof multiple identities and control multiple votes. Those identities may be created by the person who controls them or they may be hijacked from other users. This is not an issue for normal Bitcoin transactions, which do not support personal identifiers.
- Tampering. When transactions are changed after a user believes they are finalized, they have been tampered with. It is assumed that the nature of the Bitcoin blockchain will provide strong protection for completed transactions, including votes. This issue is therefore most prevalent at the software and communication levels and is not considered here.
- Repudiation. When a person rejects a transaction they have allegedly made they have repudiated that transaction. In standard Bitcoin practice, the blockchain provides strong protection against this as well. However, because the community system introduces a level of identity that is not present in the baseline Bitcoin protocol, there may be issues with repudiation here, particularly in voting.
- Information disclosure. Like repudiation and spoofing, the introduction of user identity adds complications that are not present in the original Bitcoin protocol. How much complication is introduced is determined by how much identity information is disclosed versus how much is protected, which may vary greatly between different communities.
- Denial of service. At the fund level, denial of service means that the community fund is depleted. In some communities this may be seen as only a temporary inconvenience. In others it may be a catastrophe. At some level, funds must be maintained.
- Elevation of privilege. When someone is able to access funds which they do not have the right to use, or to propose loans or open or close voting when they are not designated as having that authority, they are said to have elevated their privilege. This is very sensitive to community standards.
- Operational. In addition to the STRIDE categories described above, we use the operational category to show risks that may be aggravated or mitigated by the way community cryptocurrency features, such as voting, are designed.

4.2 The Vulnerability Matrix

Using the STRIDE+O categories, and the classifications of voting, identity or loan problems, we have developed the matrix of potential problems in designing a community currency seen in Table 1 (below). This list is not considered exhaustive but should contain the most prominent risks that must be considered when designing a currency for community use.

4.3 Mitigations

Multiple issues flow from insecure identity implementations. If a small number of people can control a large number of votes - that is more than one each - they can

Table 1. Vulnerability matrix

Category	Issue	Effect	Mitigation strategy
Identity	Too few registered users (O).	Small coalition controls funds.	None (community has failed).
	Too many registered users (O).	No quorum, no completed votes.	Adjust quorum rules to accept lower percentages of voters.
	Spoofed identities (S, R, E).	People propose and vote on loans for themselves.	Require identity confirmation.
	Abandoned IDs (O).	No quorum, no completed votes.	Require proof of activity for voting.
Voting	Voter turnout too high (D).	If people are paid to vote, this could deplete funds.	Do not pay for votes or reduce payments once quorum is reached.
	Voter turnout too low (O).	No quorum, no completed votes.	Time limit voting period and reduce or eliminate quorum rules.
	Voter turnout too low due to apathy (O).	No quorum, no completed votes, few loan proposals.	Pay for votes. May pay for the first n votes or for votes in first t minutes or choose random sample of voters to be paid.
	Voter turnout low due to confusion about issues (O).	No quorum, no completed votes, voter dissatisfaction.	Require proposal summaries; use wallets or other means to support information dissemination and debate.
Loans	Too many proposals to vote on (D).	Depleted funds, not enough votes on individual items.	Limit number of concurrent proposals or pay for votes on items that have not reached quorum.
	Falsified votes (S).	See spoofed IDs.	See spoofed IDs.
	Not repaid (O).	Funds depleted.	Garnish payments after some time limit.
	Too many loans (D).	Funds depleted.	Set a maximum level of concurrent loans and/or increase interest rates.
	Too few loans (O).	Community loses credibility.	Reduce interest rates. Pay dividends to all community members.
	Too few transactions (O).	Funds depleted.	Introduce demurrage, transaction fees or request donations.

dictate who gets loans, including steering loans to themselves and their friends. One partial solution to this is to require that the identity of new voting members be verified by some number of previously existing members. It is a partial solution because it can never be guaranteed that those who provide verification will perform due diligence, such as meeting new members face to face, or that they will not be fooled. Therefore this is only a first level of protection. Some method of revoking voting privileges should exist as well, though this could also introduce the potential for problems if not carefully handled. Methods of temporarily suspending voting rights, loan proposal and loan receipt rights may also be necessary to police issues.

Controls can also be built that take note of community participation. For example, someone who has not posted a transaction of any kind within the last 30 days could be barred from voting or from proposing loans. This should discourage the creation of membership accounts solely intended to influence votes. There is a risk with this method that too many users will be barred if transaction volumes drop too low. This likely cannot be remedied by automatically or even manually adjusting the time period to allow more to vote. When participation is too low, the remedies are social. The community needs to be encouraged to participate more.

The problem of too frequent or too large loans is more readily amenable to software controls. With guidance from the world of banking and finance, formulae can be developed to balance loans that are being repaid, those that are not and the funds available to make more loans. The performance of the loan fund can be scored and loan proposals compared with currently advisable amounts before voting is allowed.

The danger of underperforming loans in community currencies is significant, especially if the community includes assisting the poor among its values. To at least partially offset this, repayment terms can be automated, for example garnishing 1% of every transaction which the loan recipient receives until the loan is repaid. This may be more reliable than mandating manual monthly payments.

Interest rates on community loans are often low or non-existent [6]. If the loan fund is low or the proposed loan high, a prudent policy might be to charge a higher rate of interest than at other times. This would discourage depletion of the fund as well as bringing in extra revenue to rebuild it. Again, this can be expressed in software easily.

The mitigation efforts described here may be implemented in various ways to meet the needs and expectations of various communities. However, unless a community has perfect trust, the issues described must be considered and controls decided on before the currency is launched.

5 Conclusion and Future Research

It has been shown here that the configuration of a community currency is significantly different from that of the standard Bitcoin model. Its needs are more social, requiring degrees of identity and fiduciary care not found in the pseudonymous world of Bitcoin. These needs produce their own opportunities for economic engagement and community cohesion while also generating significant risks for collapse of a community using a currency with a significantly flawed design. We discussed three major vulnerability areas in the community fund of a community cryptocurrency. In almost all cases we found possible mitigation strategies that could be built into the cryptocurrency software. The operation of these mitigations in practice and ways to adapt them to community values and changing conditions are important areas for future research.

Different models of identity, including those based on a model such as Namecoin and more complex, more deterministic measures should be implemented and their risk models worked out in detail. Experimentation with different loan models, such as interest bearing, non-interest bearing and multi-party, as well as differing payback rates and garnished and voluntary payments, can yield unprecedented information about financial dynamics and human financial behavior.

Equally important as the proper balance of features are the sociological aspects of how a community works. A community cryptocurrency may be a powerful tool for advancing shared goals but financial issues may also divide people. What controls, either technological or social, exist to ensure that people perform diligently and in good faith? What are the effects of information passing on the quality and nature of votes? How does the size and frequency of loans affect members' trust in the community or its governance?

The concept of a community cryptocurrency is one that joins the financial, technological and social worlds in new ways. Designing a currency that has a reasonable chance of working is only the beginning of the effort. In this paper we have tried to explore the most important variables and potential features at a high level. There are more levels and more variables still to be considered.

References

1. About: International Journal of Community Currency Research. Retrieved September 2, 2014. (n.d.) <http://ijccr.net/about/>
2. Seyfang, G.: Tackling social exclusion with community currencies: learning from LETS to Time banks. *Int. J. Community Currency Res.* **6**(1), 1–11 (2002)
3. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted **1**(2012), 28 (2008)
4. Buterin, V.: A next-generation smart contract and decentralized application platform (2014)
5. Lietaer, B.: Complementary currencies in japan today: history, originality and relevance. *Int. J. Community Currency Res.* **8**(1), 1–23 (2004)
6. Lietaer, B., Hallsmith, G.: Community currency guide. Global Community Initiatives (2006). <http://www.lyttelton.net.nz/timebank/Community%20Currency%20Guide.pdf>. July 20 2007
7. Hess, D.J.: An Introduction to Localist Movements. American Sociological Association, Denver (2012)
8. Mascornick, J.: Local Currency Loans and Grants: Comparative Case Studies of Ithaca HOURS and Calgary Dollars (Doctoral dissertation, University of Montana) (2006)
9. Ecoffey, Brandon. Oglala Sioux Tribe surprised by MazaCoin plan. Indianz. Native Sun News, 7 Mar. 2014. Web. 2 Sept. 2014. <http://www.indianz.com/News/2014/012781.asp>
10. IrishCoin. IrishCoin.org. N.p., n.d. Web. 2 Sept. 2014. <http://irishcoin.org/irishcoin.html>
11. BitcoinForMars. Not every cause needs a coin but every planet does. (n.d.). Retrieved September 2, 2014. <http://marscoin.org>
12. Weaknesses. (2014, August 2). Retrieved September 2, 2014. https://en.bitcoin.it/wiki/Weaknesses#Attacker_has_a_lot_of_computing_power
13. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without Proof of Work (2014). arXiv preprint arXiv:10.1406.5694
14. De la Rosa, J.L., Stodder, J.: On velocity in several complementary currencies. In: 2nd International Conference on Complementary and Community Currencies Systems, The Hague (2013)
15. Williams, C.C., Aldridge, T., Lee, R., Leyshon, A., Thrift, N., Tooke, J.: Bridges into work? An evaluation of local exchange and trading schemes (LETS). *Policy Stud.* **22**(2), 119–132 (2001)
16. Shostack, A.: Threat Modeling: Designing for Security. Wiley, Indianapolis (2014)

The Bitcoin Market Potential Index

Garrick Hileman^(✉)

London School of Economics, London, UK

g.hileman@lse.ac.uk

The Bitcoin Market Potential Index (BMPI) is a new composite indicator that conceptualizes and ranks the potential utility of bitcoin across 178 countries to show which countries have the most and least potential to see bitcoin adoption. The index utilizes a data set with 40 variables grouped into the index's seven equally weighted sub-indices: technology penetration, international remittances, inflation, size of informal economy, financial repression, historical financial crises, and bitcoin penetration. Data across the different BMPI variables were first standardized as follows:

$$z = \frac{x - \bar{x}}{s}$$

Where x = each data point, \bar{x} = the average of the sample data points, s = the sample standard deviation, and z = the standardized data point. Data were also re-scaled to fit a scale of 0 to 1 as follows:

$$x_{0to1} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Where x = each data point, x_{min} = the minimum value of the sample data points, x_{max} = the maximum value of the sample data points, $x_{0\ to\ 1}$ = the normalized data point, scaled from 0 to 1.

Sub-Saharan Africa is the most fertile region for bitcoin adoption, followed by Latin America and the Post-Soviet/Communist countries. Index rankings with re-scaled data are broadly similar to standardized results. The largest change observed between the two methods was for the United States, which fell from a ranking of 5th to 72th when data were re-scaled. This change was largely due to the United States' high Bitcoin Penetration ranking and the fact that, put simply, re-scaling can reduce the effect of outliers on index rankings more than standardization (Table 1).

While the BMPI provides a useful conceptual reference for better understanding the factors that may influence bitcoin adoption it is important to acknowledge some of the index's current limitations. Specifically, due to limited data availability a number of variables that will impact bitcoin adoption are not currently included in the index (e.g., regulation).

Table 1. BMPI Top 10 Country Rankings - Standardized and Re-Scaled Data

Ranking	Country (Standardized)	Country (Re-scaled)
1	Argentina	Argentina
2	Venezuela, RB	Venezuela, RB
3	Zimbabwe	Zimbabwe
4	Malawi	Iceland
5	United States	Malawi
6	Belarus	Guinea-Bissau
7	Nigeria	Congo, Dem. Rep.
8	Congo, Dem. Rep.	Belarus
9	Iceland	Nigeria
10	Iran, Islamic Rep.	Angola

References

- Böhme, R., Christin, N., Edelman, B.G., Moore, T.: Bitcoin. *J. Econ. Perspect.* (2014) (Forthcoming: 15, 2015)
- Christin, N.: Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace (2012). arXiv preprint [arXiv:1207.7139](https://arxiv.org/abs/1207.7139)
- Cirasino, M.; How can we cut the high costs of remittances to Africa?. In: The World Bank (2013)
- Hileman, G.: A History of Alternative Currencies (2014)
- Nardo, M., Saisana, M., Saltelli, A., Tarantola, S., Hoffman, A., Giovanminni, E.: *Handbook on Constructing Composite Indicators: Methodology and User Guide*. OECD publishing, Paris (2005)
- Saisana, M., Saltelli, A., Tarantola, S.: Uncertainty and sensitivity analysis techniques as tools for the quality assessment of composite indicators. *J. Roy. Stat. Soc. Ser. A (Stat. Soc.)* **168**(2), 307–323 (2005)
- Saltelli, A., Nardo, M., Saisana, M., Tarantola, S.: Composite indicators: the controversy and the way forward. *Statistics, Knowledge and Policy Key Indicators to Inform Decision Making*, p. 359 (2005)

Cryptographic Currencies from a Tech-Policy Perspective: Policy Issues and Technical Directions

Emily McReynolds^(✉), Adam Lerner, Will Scott,
Franziska Roesner, and Tadayoshi Kohno

Tech Policy Lab and Security and Privacy Research Lab,
University of Washington, Seattle, WA, USA

Abstract. We study legal and policy issues surrounding crypto currencies, such as Bitcoin, and how those issues interact with technical design options. With an interdisciplinary team, we consider in depth a variety of issues surrounding law, policy, and crypto currencies—such as the physical location where a crypto currency’s value exists for jurisdictional and other purposes, the regulation of anonymous or pseudonymous currencies, and challenges as virtual currency protocols and laws evolve. We reflect on how different technical directions may interact with the relevant laws and policies, raising key issues for both policy experts and technologists.

1 Introduction

Bitcoin [32] and other crypto currencies have recently become a key topic of research interest for the financial cryptography community and have seen significant adoption. The research community has considered numerous aspects of crypto currencies, including: the development of new crypto currencies with different properties (e.g., [8, 31]), the measurement of existing currency deployments [30], and the uncovering of vulnerabilities in currency protocols followed by the creation of defenses (e.g., [19]). In the applied world, we have seen numerous commercial efforts to mine bitcoins, serve as Bitcoin exchanges, and provide financial tools based on bitcoins. Bitcoin has been featured regularly in the news, from the acceptance by well-known institutional investors to the scandals surrounding Mt. Gox and the Silk Road. Bitcoin has also gained a level of acceptance from traditional merchants, with leading organizations like Dell [16], Overstock.com [34], and Wikipedia [24] supporting the technology.

These incidents have given observers some insights into the legal standing of crypto currencies today, as well as the challenges to making them secure, robust, and widely-used. However, to the best of our knowledge, the public academic community has not stepped back and asked from a holistic perspective: what are the most important legal and policy issues surrounding Bitcoin and other crypto currencies, and how do those issues interact with technical design options? We seek to fill that gap here. Our team combines expertise in computer security,

cryptography, and law to evaluate key issues surrounding crypto currencies. In doing so, we hope to inform present and future crypto currency designers about key issues linking technology and policy. Given our legal background, we also write this paper with law and policy experts as an intended audience and hope that this paper can be used to inform future national and international policies.

In the following sections, we consider in depth a variety of issues surrounding law, policy, and crypto currencies—also commonly referred to by regulators as *virtual currencies*—including Bitcoin. We consider the legal requirements that exist around Bitcoin (Sect. 4), the location of where Bitcoin’s value exists physically (Sect. 5.1), the regulation of anonymous or pseudonymous currencies (Sect. 5.2), and challenges as virtual currency protocols and laws evolve (Sect. 5.3).

This paper does not intend to deep dive into *all* of the technical and legal nuances related to virtual currencies, about which entire volumes could be written. Rather, we aim to expose key issues that have yet to be addressed and technical directions. In determining what were the key issues we looked to both existing policy work on Bitcoin (e.g., [9, 10, 20]), and the central questions arising in our own discussions (see Sect. 2). For tractability, our focus is also on the legal system of a single country (the United States), though we do refer to laws and policies in other countries when appropriate.

2 Our Process

We survey our research process here, to provide a context for interpreting our results. Our team is interdisciplinary, including members trained in computer security, cryptography, and the law. In parallel with significant literature reviews (including technical academic publications, as well as governmental legal and policy determinations), we initiated our research with numerous co-located meetings in which we brainstormed important areas of technology, policy, and law that relate directly or indirectly to crypto currencies.

We observed that most of our conversations centered around questions—questions from those with legal expertise about the properties of technology, and questions from those with technical expertise about aspects of the law and policy. Since these questions can represent knowledge gaps, we considered it valuable to capture these questions—and their answers—for the greater computer security and policy communities. We therefore chose to structure this paper around those questions. We then proceeded to iterate on the questions and answers, the final form appearing in Sect. 5. We realize that the resulting discussions are not exhaustive—there are countless other questions one might ask. However, we aim to cover questions of interest to both policy experts and technologists.

3 Background: Bitcoin and Crypto Currencies

Chaum’s Electronic Cash. There is a long history of work in the cryptographic community on electronic currencies. In 1982, Chaum described an

untraceable form of electronic cash [13], and in 1988 enhanced that design to prevent double-spending [12]. The premise of this system of cash was to prevent giving the same piece of electronic cash to two people (“double-spending”) by holders of cash while also preserving the anonymity of those spenders. In Chaum’s system, a central issuer (a “bank”) participated in the protocol to issue cash and to confirm to participants that transactions were valid.

Bitcoin and Relatives. More recently, Nakamoto proposed Bitcoin [32], an electronic cash system which does away with the need for a central bank. Unlike Chaum’s electronic cash, Bitcoin is fully decentralized, with no bank acting as issuer. Instead, all transactions are completely public and can be verified by any participant. The Bitcoin protocol consists of peer-to-peer interactions between participants, which maintain a public ledger of every transaction, including both the issuing of new cash and all transfers of cash. This ledger is called the *block chain*, composed of *blocks* that include one or more transactions in a time period, where the mining reward (described below) is counted as a transaction. Not all participants in the Bitcoin network need to store the entire block chain in order to verify transactions. An individual using Bitcoin generates a public-private keypair which becomes their pseudonym in the network. A *wallet* may house one or more key pairs. Transactions in the network are identified only by these pseudonyms. Bitcoin does not directly link pseudonym to real-world identities, allowing its design to claim the possibility of some level of anonymity for its users. Yet while Bitcoin tries for anonymity, there is significant evidence that de-anonymization of its users via transactions is possible [30].

Bitcoin assumes that at least half of participants are honest, and thus that half of the computational power in the system is controlled by honest participants. Nakamoto describes this as *one-CPU-one-vote*. Adding a new transaction to the block chain ledger takes a large amount of computational work, which is performed by participants in the system known as *miners*, who must solve a computational puzzle based on a cryptographic hash before including a new transaction in the block chain. Participants only accept blocks in the chain for which the answer to the puzzle has been computed and included. As such, the length of the block chain is determined by the total amount of computational power possessed by miners in the system.

Block chain forks might exist, but the protocol is designed to recover from such forks. An attacker might spend coins and then attempt to create a new block chain which does not include those spending transactions. This is effectively a double-spending attack: the attacker spends the coin once, receives goods or services for it, and then performs an attack to form a new block chain in which the spending transaction is removed, effectively recovering the coin. To defend against fake block chain forks, participants consider the longest block chain in existence to be the correct one, and a transaction is considered recorded if a sufficient number of blocks follow it in the longest block chain. Since the protocol assumes that more than half of the computational power in the network is honest, and more computational power means a block chain will be longer, it follows that the block chain worked on by honest participants will be the longest. Attackers

need to control more computational power than all other participants combined in order to create a malicious block chain that is trusted as the real ledger.

Physical Electronic Currency. Recall that bitcoins are stored in “wallets” consisting of public-private key pairs. The power to spend the coins in a wallet is held by anyone who possesses the enclosed private key. Thus, it is possible to create physical versions of Bitcoin by writing, printing, or engraving the private key for a wallet onto a physical artifact like a banknote or a coin. The private key is often hidden in a tamper-resistant or tamper-evident manner (e.g., under a sticker), to allow the coin to transfer multiple times while providing assurance to the final recipient that the private key has not been stolen en route.

Note that a user might also make their own physical copies of their private key. For example, a person might store a key, printed on paper, in a safe. Additionally, a person might keep multiple digital copies of a private key on different storage media or devices for backup or easier usage.

Exchanges and Real-World Use. Bitcoin and several of its relatives, such as Dogecoin and Litecoin, have entered real-world use recently. Businesses with the role of converting electronic coins into accepted national currencies have sprung up and are known as *exchanges*, by analogy to ordinary currency exchanges. Exchanges (such as Coinbase) facilitate the buying and selling of real currencies for electronic currencies and vice-versa. These exchanges represent a significant level of centralization in the system, as many users wish to be able to convert between *real-world* currencies and different types of electronic cash.

Internally, exchanges typically operate by matching requests to transfer value between a pair of currencies. A percentage of matched volume is typically taken from one of two parties as revenue for the exchange. Most Bitcoin exchanges require that users transfer coins they wish to sell into a wallet controlled by the exchange. This lowers risk for prospective buyers, who now only need to trust the exchange and not also the user who previously owned the bitcoins they are buying. The risk is instead held by sellers who must trust the exchange to honor its obligations, and to protect their personal information needed for payments.

4 Analysis of Relevant Legal Contexts

The first step in any legal analysis of an emerging technology is to look to existing legal frameworks. Before examining the key tech-policy issues, in this section we begin with an overview of existing legal frameworks and current legal status of virtual currencies. While our effort in this paper is focused on virtual currencies in general, because of Bitcoin’s widespread examination and adoption, many of our examples rely on determinations specific to Bitcoin.

It is often said that technology moves faster than law, and while that can be true, laws and regulations should be designed when possible to be broad enough to cover future developments. There is currently some proposed Bitcoin-specific regulation, including in the U.S. (New York State) [33] and in France [29], but the danger is that the legislation will be out of date before it can even be fully

implemented. Examining virtual currencies as a general category, as the U.S. and European Union (EU) have done, allows laws to be more adaptable.

In the U.S., the starting point for laws and regulations is the U.S. Constitution, which includes the authority for the U.S. Congress to “coin money” and “regulate the value thereof” [1]. The Constitution provides the U.S. Congress with the ability to pass laws, which may include the creation of federal agencies with the ability to create rules. Thus, while the U.S. currently has no specific Bitcoin regulation, many federal agencies have interpreted existing regulations for their application to virtual currencies and thus Bitcoin (e.g., [14, 22, 27, 42]).

Legal Status of Bitcoin and Other Virtual Currencies. The legal status of Bitcoin and virtual currencies varies from country to country. Often Bitcoin’s legality is related to a country’s currency controls. In China, where there are strong currency controls, Bitcoin was specifically banned for financial and payment institutions; however, participating in the network and mining appears to be allowed [39]. Sometimes Bitcoin’s use is limited due to a lack of understanding. In Thailand, for example, the Central Bank halted its use after an attempt by a Bitcoin exchange to present their business to banking authorities [7]. The Thai determination was made based on the lack of applicable laws.

Meanwhile, Bitcoin is (at time of writing) legal in other countries, including the U.S. and the EU, who are working to regulate virtual currencies (e.g., [18, 22, 27]). We mention specific U.S. regulations where they apply in later sections.

Thailand’s ban may be lifted when the legislature clarifies Bitcoin’s legal standing, but some Bitcoin enthusiasts believe that the myriad of rules the U.S. government is applying may actually be more detrimental to Bitcoin’s future since, while an outright ban may be lifted, the layered regulations in the U.S. are complicated and make Bitcoin less easy to use [25]. With this legal background we turn to tech-policy issues for cryptographic currencies.

5 Tech-Policy Issues for Crypto Currencies

We now consider questions that span technology, policy, and the law. Through literature review and our discussions, we identified three areas under-addressed in previous work: understanding where the money resides; distinguishing anonymity versus pseudonymity; and issues that arise as the world evolves. These questions span multiple areas of law; our aim is not an exhaustive examination but to look at the current situation, both legal and technical, with a view towards technical mechanisms that will improve policy outcomes. We anchor much of our discussions in Bitcoin in particular due to its current popularity, but many of our analyses extend to other cryptographic or virtual currencies.

5.1 Where Is the Money?

We found the following question arise in numerous forms: Where is the money located in Bitcoin or any similar crypto currency system? This question is

relevant to understanding the laws applicable to interactions between virtual currencies and international border crossings, taxation, theft, and insurance.

From a technical perspective, it may be tempting to say that there is no physical manifestation of the money, and hence it does not exist in any one physical place at a given time. However, from a legal perspective, determining the location of the money is seen by governments and their regulators as the first step to concluding who has jurisdiction—who has the official power to make regulatory and policing decisions. Thus, though a technologist may argue that the money has no physical location, the law will likely decide that it does, and hence we explore a set of options below. Although we consider these different possibilities for the location of virtual currencies and particularly Bitcoin, we stress that not all options are equal in terms of technical or legal viability.

Location of Private Keys. Because the private key gives access to the bitcoins for transactions, one might argue that the private key location could provide the basis for legal location. However, a private key is not a singular item: unlike a safe deposit box that exists in only one location, a private key can be stored on a hard drive, in the cloud, on a piece of paper, or even memorized so as not to be written anywhere. Further, a private key could be split into multiple components that are each stored at different locations or using different storage mechanisms.

Money is with the Individual. In practice, an individual could access their bitcoins from anywhere, since all they need is their private key and Internet connectivity. In this situation, a determination that the bitcoins are where the individual is or is using their bitcoins could provide a workable foundation for application of laws. In U.S. criminal cases thus far—Silk Road/Ulbricht [5], Bitcoin Savings and Trust Ponzi Scheme [3], Faiella and Schrem [6]—the prosecution has evaluated jurisdiction based on the location of the individual and the exchange. From a technical perspective, we observe that a person might use a remote server to perform transactions, which means that their private keys may not be near them.

Location of the Exchange. While some transactions go directly between individuals, others use exchanges. If an exchange is used for Bitcoin transactions, determining the location of the bitcoins may be an easier task for jurisdictional purposes. Exchanges might hold bitcoins for you (like having an account at a bank), might facilitate buying and selling of Bitcoin (like a stock market), and even provide insurance. While exchanges began with very little regulation attached to them, currently government authorities in many countries are beginning to regulate them. In the U.S., exchanges must register as a money services business with the Department of the Treasury, requiring them to follow money transmitter laws in all of the U.S. states [22]. In 2013, Mt. Gox (prior to all of its other problems) and its U.S. subsidiary were pursued for failing to register as a money services business [4]. If bitcoins are stored in an exchange, like Coinbase or Bitinstant, because that business should be registered with a government somewhere, jurisdiction could be based on the location of the exchange.

The generation of bitcoins through the mining process is also centralized in practice in the form of *mining pools*. If the mining pool is registered as a business it would provide jurisdictional location similar to an exchange. In mining pools, users submit hashes to a centralized coordinator, who divvies earnings from the pool across all members based on work. Mining pools are typically distinct entities from exchanges, and so far have received less regulatory scrutiny, but we can expect that they could be subject to similar policies in how they pay out earnings to participants.

Money is Stored in Block Chain. One might argue that the money is also stored in all locations in which the block chain is stored, even if not all parties storing the block chain can access the bitcoins. This brings up many of the same issues governments have had with regulating conduct on the Internet. The U.S. authorities have used the Internet's ubiquitous nature to prosecute crimes federally, establishing jurisdiction with the location of the prosecutor [37]. While establishing that the money as stored in all locations the block chain is stored might seem outlandish technically, legally it could be an option.

The Legal Process. From a legal perspective, determining what constitutes the location of the money will likely be an iterative process, involving different courts that may make different decisions. In the U.S. if the courts decide in one case that the money is with the private key, another court can decide in a different case that the money is with the individual. For example, Ross Ulbricht is currently in federal court but at a district court level, which means any decisions made there do not have to be followed by other courts [5]. In the case of an appeal, decisions made by a Federal Appeals Court apply only to that court's region; cases may go to the Supreme Court when one circuit disagrees with another [40].

Case Studies. We now present several case studies in which we consider what the law is trying to accomplish by locating the money. We explore what the law currently is, what future laws might be, the technical mechanisms that should be considered for future policymaking, as well as possible directions for the technical community. We conclude the section with a discussion of what technologists might consider to impact these laws.

5.1.1 Border Crossings

Current Situation. Most countries have a requirement that monetary instruments above a certain value be declared as people cross the border. When transferring bitcoins through an exchange, most likely the exchange has established the necessary legal procedures registering either as a money services business, as is currently required in the U.S. and Canada, or meeting other government requirements. If the law determines that a bitcoin does have a location (contrary to a technical argument that no such location exists), then carrying it across a national border will have legal implications. Currently, in the United States any amount over \$10,000 must be declared on a FinCEN form 105, and the penalty for not doing so can be up to a \$500,000 fine and up to ten years in

prison [41]. These rules are in place as part of anti-money laundering measures; the implications of money laundering will be discussed further in Sect. 5.2.

Future Possibilities. There is discussion that virtual currencies are not monetary instruments. Using the IRS's determination that virtual currency is property and not currency [27], the Silk Road's accused founder argued through his attorney that Bitcoin could thus not be a monetary instrument [17]. Presently, the argument had been unsuccessful [23] but could be revisited on appeal. This may mean that future laws will clarify virtual currency's status. Already a bill has been proposed in the U.S. Congress to declare Bitcoin currency [36].

Technical Mechanisms. We now explore technical nuances that should be considered for future policymaking. Using the location of the private key for jurisdiction makes sense when the private key is stored on paper or on a hard drive in a location such as a safe. It could then be considered to be in that location rather than on the person crossing the border. However, if the private key is stored in a safe in the U.S. and another copy is stored in another country, then the authority seeking to determine its location may argue that the coins are in both locations. If the courts make such a determination, then by replicating a key in different locations, an individual may subject themselves to each of those jurisdictions. This possibility suggests that replicating keys in multiple locations (something individuals may wish to do for recovery purposes if one location's keys are destroyed) may create new legal challenges for the individual.

Another possibility is if the private key is in a password-protected format. In the U.S., it would be difficult for an authority to force individuals to reveal their password because of rights against self-incrimination; however, there are many countries where this would not be true. If threshold cryptography were used to split the private key into two parts, with each part stored on a different smartcard where both smartcards are needed to perform transactions with this private key, a government authority might simplify the situation by attributing location to the individual. Governments would likely make the same attribution—that the money is with the individual—if individuals use other sophisticated techniques to obscure (their belief of the interpretation of) the “location” of the money, e.g., by encrypting the private key with another key for which the corresponding decryption key is stored in another jurisdiction. Thus, as with some of the earlier examples, we see that technical mechanisms designed to achieve one valuable property (e.g., security) have the potential to negatively impact an individual when trying to determine the location of the user's coins.

Turning to exchanges, we observe that an exchange can be treated similar to a bank and therefore when an individual crosses a border it is as if the money is in a bank account. However, this also leads to the following scenario: A user transfers bitcoins from his private wallet managed by an exchange in Japan to purchase goods from a U.S. merchant using an exchange in Canada. If the locations of the exchanges are used as the location of the money, this process could be interpreted as him or her transferring funds from Japan to Canada—even if he or she, and the destination merchant, were in the U.S. the entire time.

5.1.2 Taxes

Current Situation. Countries have taken different stances on the status of virtual currencies for tax purposes. In the U.S., the IRS has determined that virtual currency and Bitcoin will be taxed as property [27]. This means that capital gains tax applies and even that Bitcoin mining must be reported under self-employment tax requirements. Both Brazil and Finland have also required citizens to report Bitcoin investment income as capital gains [38]. While some countries have established how virtual currencies like Bitcoin are to be taxed, the real-world application of taxation is quite complicated. In the U.S. there are rules covering different kinds of income, taxes based on where and how the income was earned, and when securities are bought and sold. There are also requirements about reporting income earned in foreign countries [26]. We do not go into detail here about forms of taxation for that reason.

Future Possibilities. While virtual currency has been considered taxable income in countries including the U.S. for years [28], the determination that it and Bitcoin is to be taxed as property is a recent development [27]. Since there is little Bitcoin-specific regulation thus far, it is possible that a legislature or central bank could still determine that Bitcoin is currency. In the U.S., legislators have begun proposing laws to classify Bitcoin as currency [33, 36]. Whether this classification would have a positive impact is unclear. If it were currency, the legal implications could be problematic because of laws in many countries establishing that only the government has the right to issue currency [38].

Technical Mechanisms. If the private key provides the legal location of the money, and the key is stored in a single location such as a safe, then determining which government's taxation rules apply might be simpler than in alternate scenarios. If, however, the location of the money is with the individual, the complex rules for an individual's income, whether earned in their country of residence or foreign location, would likely apply. Attributing the location of the money to an exchange may provide more clarity if the exchange is registered with a government, though rules about individual income taxation would likely still apply. Even if one were to argue that the money in Bitcoin is in the block chain and therefore on every node, tax authorities would likely still rely on the individual's location and self-reporting of income. In other words, technical options (like claiming that the money does not exist in any single location) cannot enable people to avoid laws like taxation, and legal determinations may not always match what (some) technologists believe to be true.

Stepping back, we find that the question of taxation is related to the previous question of “where is the money?” because the location of the money can determine the applicable jurisdiction. As with the earlier discussions, technical decisions can help simplify the determination of where the money is (e.g., only store a single copy of the private key in a single location) or complicate matters (e.g., replicate a key across multiple jurisdictions, or use secret-sharing to split a key across multiple jurisdictions so that pieces from different jurisdictions are required in order to reconstruct the key). However, policymakers may choose

specific answers for questions like “where is the money?” that are not directly correlated to the system’s technical properties. Nevertheless, we believe that it will empower technologists to understand the issues that we analyze here.

5.1.3 Theft and Fraud

Current Situation. If an unauthorized charge is made on a bank account or credit card as a result of card or identity theft, there is recourse to be had. The financial institution will typically reimburse the false charges. If consumers object to a charge on their account, banks may even reverse the charge. Today’s widely-deployed virtual peer-to-peer currencies do not provide this option. Once a Bitcoin transaction is confirmed into the block chain it is not reversible (under normal functioning of the system, with an honest miner majority). Proving bitcoins were stolen may eventually lead to the government seizing them from the guilty party, when technically possible, e.g., when the government obtains the corresponding private keys. If the thieves have “lost” their private keys, however, then the money may be lost from the system entirely.

Future Possibilities. Exchanges may prove to be a policy-based answer to providing recourse for theft and fraud victims. Depending on their terms of service and the country they are registered in, an exchange could provide similar services to those financial institutions currently perform. For example, exchanges could build into their model the cost of lost bitcoins, just as banks do not often recover the money they refund customers but view it as a cost of doing business. In other words, exchanges could absorb the cost of lost bitcoins rather than reversing transactions. There is also a developing market for insurance. Most insurance protections against crime do not protect against bitcoin theft, though certain exchanges (e.g., Circle, Coinbase, Elliptic, and Xapo) do provide some form of insurance. Insurance may be an easier solution for bitcoin theft, compared to pursuing criminal charges and possible later seizure from the guilty party.

Technical Mechanisms. There are several technical options for responding to theft that could be implemented through protocol changes. As one technical direction, though not one that we necessarily advocate for, would be to modify the Bitcoin protocol such that coins can be tagged with a set of allowable target addresses (or domains) for transactions. Thus, the system could ensure that bitcoins never leave a set of authorized exchanges, and these exchanges could have an agreement to (effectively) revoke transactions. Alternately, it might be possible to modify the Bitcoin protocol and allow coins to be “tainted.” For example, trusted authorities could somehow flag bitcoins to indicate that they are “dirty.” Miners or individuals could choose to ignore dirty money and any money that derives from that money. Such a protocol modification would allow a trusted party to revoke transactions—a change that may be met with resistance by some members of the cryptographic currency community, but may help prevent criminals from spending stolen money.

5.1.4 Technical Directions and the Law

Above we reflected on the legal context surrounding Bitcoin and other cryptographic currencies and have discussed the interplay between technology and policy. In doing so, we raised potential directions for additional exploration within the technical community. However, we stress that while technical innovations may affect legal decisions—that is, technical architectures *can* affect policies—technical innovations cannot prevent prosecution. For example, suppose someone uses threshold secret sharing with a private keys in an attempt to avoid laws in a particular jurisdiction. If the person did something illegal, the courts would still try to find a way to prosecute, even if the details of the case (e.g., where the money is) may not directly relate to technical properties of the system.

5.2 What About Anonymity and Pseudonymity?

There are a number of public misconceptions surrounding anonymity in virtual currency. Part of legislators' concerns over money laundering is their belief that cryptographic currencies provide anonymity. However, as some Bitcoin experts would point out, the blockchain's public nature makes Bitcoin pseudonymous rather than anonymous. Providing a better understanding of Bitcoin's status as pseudonymous is key to successful policymaking.

Bitcoin and other crypto currencies explore a new space in financial privacy compared to other technologies like cash, checks, and credit cards. One key difference is that the block chain is public, though identities can be pseudonyms. Another key difference is that virtual currencies enable the quick transfer of large sums of money outside of conventional systems (e.g., without banks).

While in Bitcoin it may not always be possible to use the public block chain to link transactions with the involved parties, it can sometimes be possible [30]. While Bitcoin entities can use techniques to make linking more difficult, like using mixers (aka, tumblers or laundries) or throw-away public-private key pairs, the mere potential for anyone with access to the public block chain to link transactions with individuals raises challenges for the finance sector. Similarly, the potential for individuals to mask their identities and transfer large sums of money also creates challenges for the financial sector. We explore these issues, as well as the implications of strongly anonymous crypto currencies, here.

On the Potential for Anyone to Link Some Transactions to Parties. Bitcoin's public block chain has proven to allow transactions to be traceable both by academics and authorities. Meiklejohn et al. [30] studied methods that could be used to re-identify the groups behind Bitcoin transactions.

In traditional banking, there are privacy requirements regarding what information banks can share with other groups and how they must preserve an individual's privacy [11]. These provisions have yet to be applied to virtual currencies and Bitcoin specifically. The nature of the protocol likely means that they could not apply to the protocol itself but since exchanges have to register in the same way that financial institutions do [22], they may eventually be subject to the same compliance requirements.

There exist numerous other examples for which the ability to link transactions with individuals might lead individuals to not use (accept or send) bitcoins. For example, consider political party donations. The U.S. Federal Election Commission has decided that campaigns may accept bitcoins, but that such a contributor must provide his or her name, physical address, and employer, and affirms that the contributed bitcoins are owned by him or her and that the contributor is not a foreign national [21]. These requirements could link them to that wallet and/or transaction, as well as enable the tracing of others.

On the Potential for Parties to Mask their Identities. One of the central concerns about virtual currencies for governments is the potential for money laundering through the potential for anonymity that Bitcoin and other virtual currencies provide [10,38]. Money laundering is generally defined as activities and financial transactions that are undertaken to hide the source of the money. Part of the money laundering concern is based in a misunderstanding of Bitcoin's status as an anonymous exchange. Since Bitcoin transactions have proven to be traceable, it is not strongly anonymous but rather pseudonymous.

On Strongly Anonymous Crypto Currencies. Zerocoin [8,31] poses a more realistic anonymous money laundering threat. Zerocoin uses the same distributed ledger as Bitcoin, but encodes transactions such that they do not reveal how much currency was transferred or publicly reveal the public keys of the sender and receiver, while still allowing for distributed verification of the ledger. While transfer in and out of the Zerocoin system in practice continues to use centralized exchanges which could be subject to regulation, transactions within the system cannot be monitored in the same way as with Bitcoin, and it is not possible for an external observer to determine how much currency an individual wallet possesses without access to the private key.

As currently designed, Zerocoin would validate regulator concerns over the money laundering threat of virtual currencies. The development of Zerocoin, if its use becomes widespread, could have a large impact on the developing regulation around Bitcoin and virtual currencies. We stress, however, that there are both advantages and disadvantages with strongly anonymous cryptographic currencies. As noted above, increased anonymity could facilitate adoption by financial institutions given current laws about bank transaction privacy, and anonymity could also facilitate adoption in contexts where privacy is highly valued (e.g., paying for certain types of medical care).

Technical Mechanisms. While privacy advocates may disagree, from an intellectual perspective, the law and policy community may be interested in the existence of escrowed anonymity systems. For example, would it be possible to create a cryptographic currency that is strongly anonymous unless a quorum of trusted third parties agree to de-anonymize a set of transactions? Or would it be possible to create a cryptographic currency that is strongly anonymous for n -years, after which the strong anonymity of each transaction melts away? For example, each transaction could be encrypted in a provably verifiable way to a time-specific public key of a trusted authority (or set of authorities). Those

authorities could commit to releasing the corresponding private keys after the appropriate time has lapsed. The resulting protocol would not require trusting the authorities to assist in the preservation of the integrity of the network, but would trust the authorities not to de-anonymize transactions early. We encourage the technical community to explore these and similar dimensions as well.

5.3 What Happens as the World Evolves?

The world is not static. New crypto currencies will be developed, the Bitcoin protocol will evolve, and the uses of Bitcoin and the Bitcoin infrastructure may evolve as well. We now shift our attention to policy and technical issues to consider as this evolution takes place. We begin with a short detour: a discussion of the Computer Fraud and Abuse Act.

Computer Fraud and Abuse Act. In the U.S., the Computer Fraud and Abuse Act (CFAA), passed in 1986, assesses whether someone knowingly accessed a computer without authorization or exceeded their authorized access [2]. Prosecutors in the U.S. have used a broad interpretation of these clauses in criminal cases. Many countries, including Brazil, the UK, and others, have equivalent laws [15], but the U.S. is one of the prominent prosecutors. Though the CFAA may seem broad and unlikely to apply to virtual currencies, recent history has shown that it is often applied to most criminal cases involving computers.

While U.S. courts have not yet specifically ruled on the CFAA’s application to virtual currencies or Bitcoin, the alleged creator of the Silk Road has been charged with violations of CFAA [5]. Because CFAA covers unauthorized access, there are a number of possible applications to protocol attacks.

The 51 % Attack, and Other Attacks. It is well known that an adversary with more than 50 % of the resources in the Bitcoin mining network could cause attacks against the integrity of the block chain. Other attacks also exist, such as selfish mining [19]. Suppose that a party—such as a mining pool—can put itself in a position to mount one of these attacks, e.g., by controlling 51 % or more of the resources in the mining network. Depending on the method of attack and its impact, it is possible that the operators could be charged with CFAA violations or their local equivalent. While technically we desire a cryptographic currency that is strong enough to resist reasonable technical attacks, we find the conclusion in this paragraph valuable because it means that even if an organization develops the capability of compromising the integrity of the block chain, there may be legal impediments for them making use of those attack capabilities.

Vulnerabilities. When a vulnerability is discovered in a popular product produced by a major manufacturer—such as a browser or an operating system—it is generally clear who the responsible party is for fixing the vulnerability. If a vulnerability is found in a particular Bitcoin mining or client software package, then it is also clear who should be responsible for fixing the vulnerability. But suppose that the vulnerability is uncovered in the protocol itself. There would need to be a process to ensure that the protocol—as well as all relevant implementations—can be updated quickly after a vulnerability is discovered. This means having

a process in place in advance of any vulnerability discovery. As Bitcoin (or any other cryptographic currency) becomes more popular and accepted by societies and governments, we argue that there becomes an increasing need for a clear process for protocol updates. Whereas there may already be a set of core developers trusted with the process today, the acceptable trust model may change if a government determines that the protocol constitutes a currency. As a point to consider: would laws speak to specific protocol versions, or protocol update processes?

Because of the challenges with protocol updates, we propose a technical stop-gap mitigation technique which could potentially be incorporated into the protocol: If people own bitcoins under version M of the protocol, and version M is found to be insecure, those people could have the capability to freeze their bitcoins (so that they cannot be spent) until version N of the protocol, $N > M$.

Illegal Content in the Block Chain. Bitcoin miners store copies of the block chain, and they may have some expectations for what will be stored in the block chain. For example, they may have the expectation that only transactions are stored in the block chain. There are already, however, many non-transaction related images and text embedded in the block chain [35]. Suppose that some malicious party, Mallory, inserts illegal content in the block chain (such as classified government documents, certain types of pornography, and so on). Mallory's actions could cause Alice's and Bob's computers to store copies of that illegal data, without Alice's and Bob's knowledge or consent. Alice and Bob might face liability in criminal charges though they did not put the content there.

While one might evaluate whether the license agreements for some (although perhaps not all) Bitcoin clients and miners clarify that arbitrary content may be stored in the block chain, raising the awareness of such potential does not mean that illegal or inappropriate content might not be inserted into the block chain. A separate question therefore arises: could Mallory's actions ultimately lead to Bitcoin becoming illegal in some jurisdictions? For example, what would the legal implications be if any node wishing to store the entire block chain must, by definition, also store certain types of illegal pornography? (The definition of legal content may vary between legal jurisdictions.) Similarly, suppose that someone inserts classified U.S. documents or corporate intellectual property into the block chain. Could these possibilities lead to Bitcoin becoming too risky for citizens to use or could these possibilities result in it becoming illegal to run a full Bitcoin node? It may be possible to hold those storing the node legally responsible for the block chain content because it is on their server or computer.

While the use of a central authority may be antithetical to some of the principles underlying Bitcoin, a future version of the protocol (or a similar protocol) might allow a central authority (or threshold set of trusted authorities) to remove specific blocks from the block chain. In the new cryptographic currency, it may be possible to use cryptographic techniques, under some trust model, to (1) allow the removal of the relevant block but create a new block that would (2) allow a verifier to verify that only certain transactions were removed or obfuscated and also (3) allow a third party to verify the details of the financial transactions

associated with the illegal content (so that any money transferred with the illegal content is not lost). We leave as an open problem whether it might be possible to achieve these properties in a new cryptographic currency, or to otherwise develop a currency that would support the removal of illegal content under a reasonable threat model.

Excessive Content in the Block Chain. Mallory might also attempt to excessively grow the size of the block chain, thereby depleting the resources of those nodes storing the entire block chain. In doing so, Mallory's actions will result in the consumption resources on those nodes without those nodes' consent. Depending on the agreements and contracts signed, Mallory could face liability in a tort lawsuit for damages and cost reimbursement.

Protocol Updates. Society, corporations, and individuals may become more dependent on Bitcoin (or other crypto currencies) over time. Bitcoins are already accepted by major online organizations like Dell and Wikipedia, and the U.S. government has already determined that (for now) Bitcoin should be treated as a commodity. In the future, there is a possibility of Bitcoin (or a derivative) becoming a currency. A question arises: what happens if the Bitcoin protocol is subsequently updated? This question is important because a protocol update could introduce a vulnerability or change the economics of participation in the Bitcoin ecosystem. In the latter case, we argue that for the United States, the U.S. National Institute of Standards and Technology (NIST) may be the appropriate body for specifying the details of the cryptographic protocol. Should NIST ever be put in a position of being asked to formalize the specification of a cryptographic currency, we suggest that NIST consider a competition-like process, similar to the process it used to select AES and SHA3. The resulting currency may be similar to Bitcoin but may not be Bitcoin if the current Bitcoin developers do not agree to transferring responsibility to NIST.

Digital Divide. A virtual currency, almost by definition, suggests a technical requirement for participation in the ecosystem. While there are corner cases that might seem like exceptions (e.g., printed Bitcoin coins), individuals operating in those cases can still benefit from technology (e.g., to use a computer to verify the integrity of a printed coin). Virtual currencies thus have the potential to amplify the digital divide—to widen the gap between those who have technologies (and hence the ability to fully participate in the crypto currency ecosystem) and those who do not. A policy implication is that it may be unlikely for a virtual currency like Bitcoin to ever be the only currency accepted in a particular jurisdiction. A technical implication is the opportunity for a low cost, easily accessible Bitcoin hardware implementation which would assist in the narrowing of the gap—by making Bitcoin technology accessible to all (possibly with a government subsidizing the (low) cost of the hardware). Until such time, we believe that a crypto currency will likely not be the *only* currency in a jurisdiction.

6 Conclusion

The interactions between technology, policy, and law for crypto currencies such as Bitcoin are often complicated and nuanced. We have considered key issues, including the physical location of the value in a crypto currency, the interaction between regulation and anonymity or pseudonymity in a virtual currency, and challenges that arise as the world evolves. While technical architectures can affect laws and policies, and vice versa, we also observe that—contrary to what technologists might prefer—laws may not necessarily match a technologist’s expectations. We explore the interplay between the law, policy, and technical mechanisms in our analysis.

We believe that the results of our analysis will be beneficial to law and policy makers. As a consequence of our interdisciplinary investigations into crypto currencies, we also proposed several directions for the technical community. Some of the directions we propose may be compatible with Bitcoin’s underlying tenets such as distributed control, whereas others may require more centralization. In suggesting technical directions for future research and innovation, we do not mean to also suggest that any one directions is more valuable or appropriate than another; we leave that decision to the reader and the technical community based on their individual values and interests. The directions that we propose are possible approaches to address current legal and policy challenges, and may possibly also help shape future policies.

Acknowledgements. We thank Jeff Haley, the members of the University of Washington Tech Policy Lab, and the members of the University of Washington Security and Privacy Research Lab for valuable discussions, and we thank our anonymous reviewers for feedback on an earlier version. This work was supported in part by NSF Award CNS-0846065, by the Short-Dooley Endowed Career Development Professorship, and by the University of Washington Tech Policy Lab.

Appendix: Disclaimer

Since this effort involves the collaboration of individuals in both the law and technology fields, we must give this disclaimer: This analysis is for informational purposes only and does not constitute legal advice.

References

1. U.S. Constitution, Article I, Section 8
2. U.S. Title 18, Section 1030 (Computer Fraud and Abuse Act)
3. Securities and Exchange Commission vs. Trendon T. Shavers and Bitcoin Savings and Trust, July 2013. <http://www.sec.gov/litigation/complaints/2013/comp-pr2013-132.pdf>
4. Seizure Warrant for Mutum Sigillum, a subsidiary of Mt. Gox, May 2013. <http://cdn.arstechnica.net/wp-content/uploads/2013/05/Mt-Gox-Dwolla-Warrant-5-14-13.pdf>

5. United States vs. Ross William Ulbricht, October 2013. <http://www.popehat.com/wp-content/uploads/2013/10/UlbrichtCriminalComplaint1.pdf>
6. United States vs. Robert M. Faeilla and Charlie Shrem, January 2014. <http://www.justice.gov/usao/nys/pressreleases/January14/SchremFaiellaChargesPR/Faiella>
7. Bangkok Pundit: Has Bitcoin really been banned in Thailand?, July 2013. <http://asiancorrespondent.com/111332/has-bitcoin-been-banned-from-thailand/>
8. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: decentralized anonymous payments from Bitcoin. In: IEEE Symposium on Security and Privacy (2014)
9. Brito, J., Castillo, A.: Bitcoin: a primer for policymakers. Mercatus Center, George Mason University, August 2013. <http://mercatus.org/publication/bitcoin-primer-policymakers>
10. Bryans, D.: Bitcoin and money laundering: mining for an effective solution. Indiana Law J. **89**(1), 441–472 (2014)
11. Bureau of consumer protection. In Brief: The Financial Privacy Requirements of the Gramm-Leach-Bliley Act, July 2002. <http://www.business.ftc.gov/documents/bus53-brief-financial-privacy-requirements-gramm-leach-bliley-act>
12. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
13. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology, CRYPTO 1982 (1982)
14. Consumer Financial Protection Bureau: risks to consumers posed by virtual currencies, August 2014. http://files.consumerfinance.gov/f/201408_cfpb-consumer-advisory_virtual-currencies.pdf
15. CyberCrime Law: Cybercrime laws from around the world. <http://www.cybercrimelaw.net/Cybercrimelaws.html>
16. Dell: Dell now accepts Bitcoin (2014). <http://www.dell.com/learn/us/en/uscorp1/campaigns/bitcoin-marketing>
17. Dratel, J.L.: Memorandum of Law in Support of Defendant Ross Ulbricht's Pre-Trial Motions Challenging the Face of the Indictment, April 2014. <http://www.wired.com/wp-content/uploads/2014/04/Ulbricht3.pdf>
18. European Central Bank: Virtual Currency Schemes, October 2012. <http://www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemes201210en.pdf>
19. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography and Data Security (2013)
20. Fairfield, J.: Smart Contracts, Bitcoin Bots, and Consumer Protection. Washington & Lee University Law Review Online, September 2014. <http://lawreview.journals.wlu.io/smart-contracts-bitcoin-bots-and-consumer-protection>
21. Federal Election Commission: FEC Memorandum, May 2014. http://www.fec.gov/agenda/2014/documents/mtgdoc_14-24-b.pdf
22. Financial Crimes Enforcement Network: Guidance FIN-2013-G001: Application of FinCEN's Regulations to Persons Administering, Exchanging, or Using Virtual Currencies, March 2013
23. Greenberg, A.: Forrest denial of defense motion in silk road case, July 2014. <http://www.scribd.com/doc/233234104/Forrest-Denial-of-Defense-Motion-in-Silk-Road-Case>
24. Gruwell, L.: Wikimedia foundation now accepts Bitcoin, July 2014. <http://blog.wikimedia.org/2014/07/30/wikimedia-foundation-now-accepts-bitcoin/>
25. Hill, K.: 21 Things I Learned About Bitcoin Living On It A Second Time, May 2014. <http://www.forbes.com/sites/kashmirhill/2014/05/15/21-things-i-learned-about-bitcoin-living-on-it-a-second-time>

26. Internal Revenue Service. <http://www.irs.gov>
27. Internal Revenue Service: Notice 2014-21, March 2014.<http://www.irs.gov/pub/irs-drop/n-14-21.pdf>
28. Internal Revenue Service: Tax Consequences of Virtual World Transactions, August 2014. <http://www.irs.gov/Businesses/Small-Businesses-&-Self-Employed/Tax-Consequences-of-Virtual-World-Transactions>
29. Marini, P., Marc, F.: La Regulation a L'epreuve de L'innovation: Les Pouvoirs Publics Face au Developpement des Monnaies Virtuelles. French Senate, July 2014. http://www.bitcoin.fr/public/divers/docs/Rapport_de_la_commission_des_finance_du_Senat.pdf
30. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Internet Measurement Conference (2013)
31. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from Bitcoin. In: IEEE Symposium on Security and Privacy (2013)
32. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). <https://bitcoin.org/bitcoin.pdf>
33. New York State Department of Financial Services: BitLicense Framework: Title 23. Department of Financial Services Chapter I. Regulations of the Superintendent of Financial Services Part 200. Virtual Currencies, Jul 2014. <http://www.dfs.ny.gov/about/press2014/pr1407171-vc.pdf>
34. Overstock.com: Bitcoin on overstock.com (2014). <http://www.overstock.com/Bitcoin>
35. Shirriff, K.: Hidden surprises in the Bitcoin blockchain and how they are stored: Nelson Mandela, Wikileaks, photos, and Python software, February 2014. <http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html>
36. Stockman, S.: Stockman plans to introduce the “virtual currency tax reform act”, April 2014. <http://stockman.house.gov/media-center/press-releases/stockman-plans-to-introduce-the-virtual-currency-tax-reform-act>
37. Tavakoli, Y., Yohannan, D.: Personal Jurisdiction in Cyberspace: Where Does It Begin, and Where Does It End? Intellectual Property and Technology Law Journal, January 2011. <http://www.kelleydrye.com/publications/articles/1431.pdf>, http://www.kelleydrye.com/publications/articles/1431/_res/id=Files/index=0/Personal%20Jurisdiction%20in%20Cyberspace_IP%26Tech%20Law%20Journal_January2011.pdf
38. The Law Library of Congress, Global Legal Research Directorate Staff: Regulation of Bitcoin in Selected Jurisdictions, January 2014
39. The People's Bank of China and Five Associated Ministries: Prevention of Risks Associated with Bitcoin, December 2013. <https://vip.btcchina.com/page/bocnotice2013>
40. United States Courts: courts of appeals. <http://www.uscourts.gov/FederalCourts/UnderstandingtheFederalCourts/CourtofAppeals.aspx>
41. U.S. Department of the Treasury: report of international transportation of currency and monetary instruments. <http://www.fincen.gov/forms/files/fin105.cmir.pdf>
42. U.S. Securities and Exchange Commission: investor alert: Bitcoin and other virtual currency-related investments, May 2014). <http://investor.gov/news-alerts/investor-alert-bitcoin-other-virtual-currency-related-investments>

Blindcoin: Blinded, Accountable Mixes for Bitcoin

Luke Valenta¹(✉) and Brendan Rowan²

¹ University of Pennsylvania, Philadelphia, PA, USA

lukev@seas.upenn.edu

² University of Maryland, College Park, MD, USA

browan@cs.umd.edu

Abstract. Mixcoin is a Bitcoin mixing protocol proposed by Bonneau *et al.* which provides strong accountability guarantees [13]. However, in the Mixcoin protocol, the mapping from a user’s input to output address is visible to the mixing server. We modify the Mixcoin protocol to provide guarantees that the input/output address mapping for any user is kept hidden from the mixing server. In order to achieve this, we make use of a blind signature scheme [14, 23] as well as an append-only public log. The scheme is fully compatible with Bitcoin, forces mixes to be accountable, preserves user anonymity even against a malicious mix, is resilient to denial of service attacks, and easily scales to many users.

1 Introduction

Bitcoin is a decentralized electronic cash system that has become increasingly popular since its introduction in 2008 [27]. The base unit of currency, the *bitcoin* or *BTC*, is defined in terms of *transactions*, which are transfers of BTC from a sender address to a receiver address, where addresses are simply public keys. All transactions are stored in a public ledger, the *block chain*. Since all transactions are public, user anonymity in Bitcoin relies on pseudonyms. There has been much work on de-anonymizing users in the bitcoin block chain by linking together addresses [10, 22, 26, 28, 30, 31]. If any one of the linked addresses can be mapped to the true identity of a user, then all past and future transactions involving the linked addresses can be associated with that user, compromising their anonymity. While a discussion of the costs and benefits of anonymity is beyond the scope of this work, there is much debate over the importance of this property [5, 20, 29].

1.1 Mixing Services

To alleviate this risk of deanonymization in Bitcoin, one can use a *mixing service* or *mix*. Mixes for anonymous communication were originally introduced by Chaum [15], but many recent services have adapted this idea to the financial

B. Rowan—This work originated as a project in a computer networks course at the University of Maryland.

setting. These services allow users to exchange their bitcoins for “clean” bitcoins that—within some anonymity set—cannot be linked to their current addresses and identities by an adversary (See Sect. 4.1). To participate in a mixing operation, a user supplies some amount of bitcoins from an input address, and specifies an output address that they wish these bitcoins to end up in eventually. Although many such mixing services exist, current designs lack certain important features. We now state properties that we believe an ideal system should offer.

- *Accountability*. When a user sends funds to the mix, they should be confident that if a theft occurs, they can present a proof of the mix’s misconduct.
- *Anonymity*. The user should be the only entity that knows the mapping from their input address to their output address.
- *Resilience to Denial of Service Attacks*. The system should not be vulnerable to attacks by a small number of malicious parties that could potentially DoS the exchange [24, 34].
- *Scalability*. The system should be efficient enough to be able to scale to a large number of users and large anonymity sets.
- *Incentive for Participation*. There should be a mechanism for the mix to collect mixing fees fairly that will incentivize it to provide the service. On the user end, the service should come at a reasonable cost and should be convenient, so that a large number of users participate.
- *Backwards-Compatibility*. The system should be compatible with the current Bitcoin system to allow for practical integration.

1.2 Current Bitcoin Mixing Services

Mixcoin. Mixcoin achieves all of the above properties, except for anonymity against a malicious mix [13]. In this protocol, a mix has access to the mapping from a user’s input to output address and can store this information. At any point in the future, the mix could reveal this information and compromise the anonymity of its users.

CoinJoin. CoinJoin is a decentralized protocol in which all users must sign a joint transaction [24]. Users remain anonymous and their funds cannot be stolen, since a user will only provide its signature if they agree to the transaction. However, CoinJoin is vulnerable to a DoS by a single user if they refuse to provide a signature during the signing round of the protocol. Thus, Coinjoin is not resilient to misbehaving users.

CoinShuffle. The protocol CoinShuffle is built on CoinJoin [32]. CoinShuffle is decentralized and uses a multiparty sorting protocol [34] to achieve anonymous mixing. It ensures that a joint transaction will eventually go through by including a blaming process in which misbehaving users can be eliminated from future mixing attempts by the honest participants. There is no financial commitment required (i.e. mixing fees) for a user to participate in or DoS a round of the protocol, so a large-scale Sybil attack [19] would be a cheap way to delay a round of mixing. Overall, this seems to be a promising approach.

CoinSwap/Fair Exchange. These protocols allow users to exchange coins anonymously with no risk of theft [7, 11]. However, it is not clear how mixing fees could be collected in an anonymous manner to incentivize the mix to provide the service.

Altcoins. Another way to mix one’s bitcoins is to exchange them for some alternate currency, or *altcoin*, and at some later time exchange the altcoins back for bitcoins. Some protocols designed for this purpose are Zerocoin and Zerocash [12, 25]. These techniques can achieve strong anonymity properties, but require additional infrastructure or modifications to the bitcoin protocol.

Deployed Services. Several mixing services that are used in practice have no provable guarantees of anonymity or theft protection [1–3, 8] For example, some Bitcoin Fog users claim that their coins were stolen by the service [4], and Möser *et al.* are able to link input/output transactions in the graph of BitLaundry [26].

1.3 Our Contribution

We modify the Mixcoin protocol to prevent the mix from learning the input/output address mappings of participating users. The Mixcoin authors, as well as [24], posit that such a scheme could be possible using blinded tokens as described in Chaum’s original digital cash scheme [14]. We show that this is indeed possible and present a protocol that achieves this goal, and at the same time preserves the accountability property of Mixcoin and the mechanism for collecting fair, randomized mix fees. The main modifications that we make are the introduction of an append-only public log that is used to keep the mix accountable, and the utilization of a blind signature scheme to hide the mapping between a user’s input and output addresses from the mix.

Our proposed system meets all of the above goals for a mixing service. For accountability, we use a warranty scheme that allows the user to provide evidence against the mix if it misbehaves, similar to [13]. Anonymity against the mixing service is provided by using a blind signature scheme to hide the input/output address mappings of participants. The system is resilient to DoS attacks by a single user refusing to sign a joint transaction, as certain schemes are susceptible to [24, 34]. This is possible since the mix deals with each user individually, and can exclude users from the exchange on a case-by-case basis. The system is scalable since it does not require any computationally complex cryptography as do some other systems [34]. The scheme employs fair, randomized mixing fees to incentivize mixes to provide the service to users at a reasonable cost and within a relatively short timespan. We expect the mixing of a single “chunk” of coins to take on the order of a few hours, and multiple chunks can be mixed simultaneously. Finally, the system is backwards compatible with Bitcoin, requiring no changes to the current protocol.

2 Background

2.1 Mixcoin Summary

We give a summary of the properties and contributions of Mixcoin.

Chunk Size. Each user sends a fixed quantity v of bitcoins to the mix. The intuition behind this is as follows: suppose some amount X BTC is transferred to some output address. Then, it is easy to link this output address to the set of all input addresses that spent X BTC within that particular time interval. We want this set to be as large as possible, since “anonymity loves company” [17].

Accountability. Mixcoin forces mixing servers to be accountable for their actions. That is, when a user sends funds to a mix, they are provided with a warranty such that, in the event that the mix steals the user’s coins, the user can present proof (verifiable by any party) of the mix’s cheating. This warranty consists of a signed agreement from the mix that if the user pays funds to an escrow address specified by the mix by a certain time, then the mix will transfer an equal amount of funds to the output address specified by the user before an agreed-upon deadline.

Sequential Mixing. In the Mixcoin model, the mixing server learns the input/output address mapping, and must be trusted to keep this information private. In order to remain anonymous in the event that the mix is compromised, users are encouraged to mix coins through multiple rounds of independent mixes, which would be costly in terms of both time and mixing fees. Our proposed changes obviate the need for these extra rounds of mixing, since the mixing servers are blinded to the input/output address mappings.

Mixing Fees. Another important contribution that Mixcoin makes is the introduction of *randomized mixing fees*. The authors argue that the collection of mixing fees incentivizes mixes to act honestly. If the mixing fees are sufficiently high, then a rational mix looking to maximize profits will not risk cheating, lest they get caught. A warranty proving their dishonesty would damage their reputation and hurt their business model. Furthermore, the Mixcoin authors argue that the strongest anonymity properties are achieved when mix fees are *all-or-nothing*, instead of some fixed rate per chunk. In essence, rather than keeping ρ fraction of each chunk, the mix will keep an entire chunk with ρ probability. To determine which of the inputs to keep as a mixing fee, the mix calculates a **Beacon** function [16], which depends on a public source of randomness (e.g., future blocks of the Bitcoin block chain). The function maps a random nonce (unique to each of the inputs) to a value in the interval $[0, 1]$. If this output falls below ρ , the mixing fee rate, then the mix claims the corresponding coins as its mixing fee. According to the Mixcoin paper, a typical value for ρ might be less than 0.01, although this will depend on the market.

2.2 Blind Signatures

The idea of blind signatures originated with Chaum in 1983 [14]. In [23], Fuchsbauer gave the first efficient implementation of a blind signature scheme with round-optimal issuing [21], meaning that only one message from the user and one message from the signer is required for the user to obtain the signature. A blind signature scheme works as follows: a user wishes to obtain a signature on some message without the signer knowing the message contents. To achieve this, the user computes some *commitment* function (which can be thought of as encryption) on the message and sends that along with a *randomization* of the message to the signer. The signer then makes a “pre-signature” and sends it back to the user. From this, the user can compute an actual signature on the message by adapting the randomness. The blind signature is then a proof of knowledge of a signature on the message. The scheme achieves the following requirements: *Blindness*: Given a set of blind signatures, the signer cannot relate the signatures to their issuings. *Unforgeability*: Given a set of blind signatures, an adversary cannot compute a valid signature on a new message.

3 Blindcoin Description

3.1 Model

In addition to the two entities of the Mixcoin protocol—the user and the mixing server—our protocol requires another entity, the public log. We summarize the entities present in the system below:

Public Mix. The public mix, M , is assumed to have a long standing public key M_{pub} and associated private key M_{priv} , used for signing. The model assumes that there are many such mixes that will compete. Mixes that have poor reputations will be less likely to be chosen by users, since users want to reduce their likelihood of being cheated. Thus, a mix is incentivized to participate in the protocol without allowing any proof of cheating to be made public.

The User. The user, A , is a party that has an amount of Bitcoins in an address k_{in} (possibly linked to their true identity) and wishes to transfer the coins to a different address k_{out} , such that it is hard for any adversary to link the addresses k_{in} and k_{out} . The user is also able to post anonymously to the public log with an alternate identity, A' . This could be achieved through Tor [18], for example. The user must be careful not to allow A and A' to be linked in any way.

Public Log. The public log is a public, append-only log used for third party verification purposes. If a party breaches protocol, the public log will contain enough information to incriminate the misbehaving party. Anyone can post to the log, and messages can never be erased once they are posted. Each message is also associated with a timestamp. One possible way to implement this would be within the Bitcoin block chain. To send a message, the sender could just transfer a small amount of BTC to one or more addresses corresponding to the

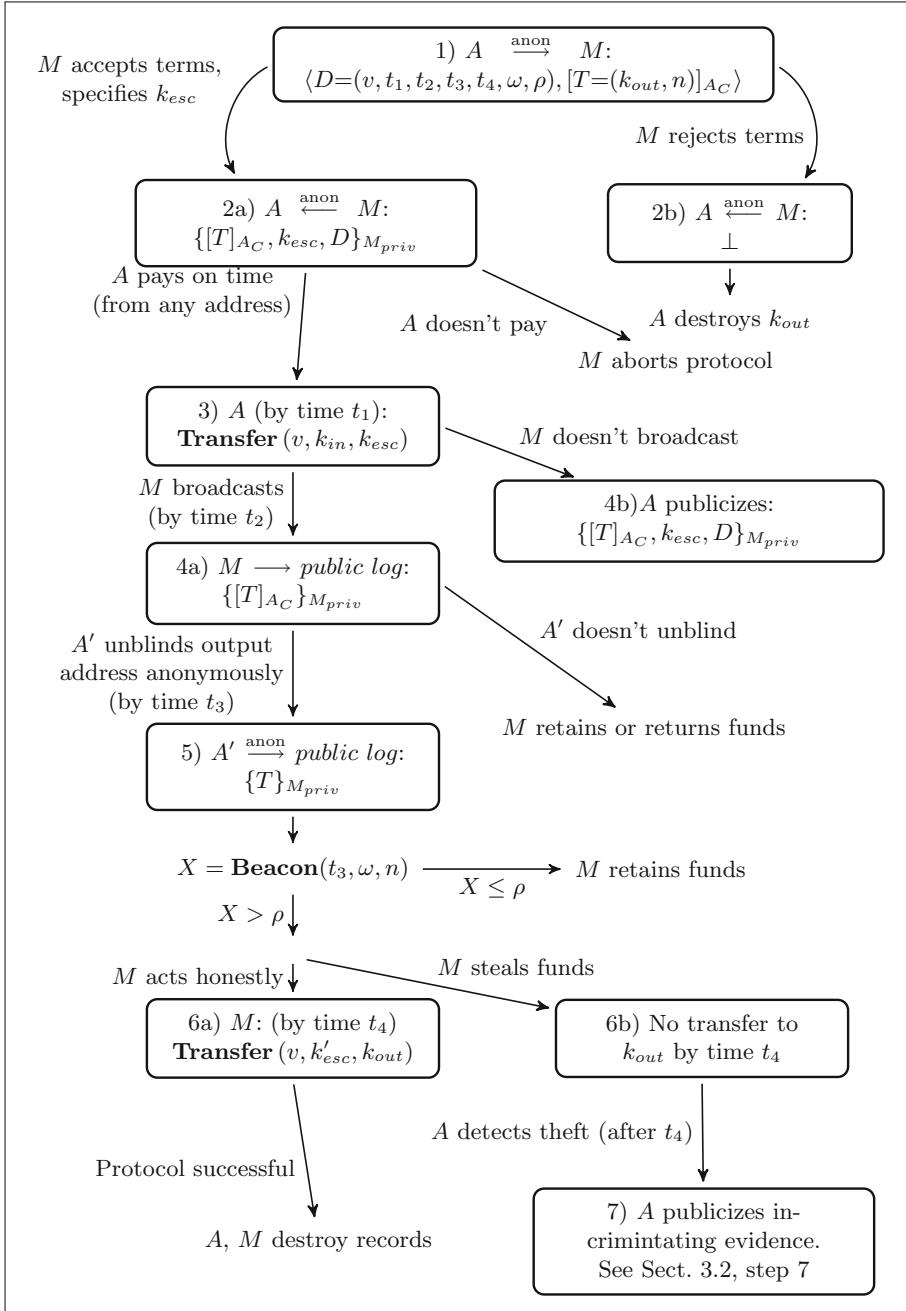
message. Of course, if the user wishes to post to the log anonymously, the coins must be sent from an address that cannot be linked back to the user's identity. An example of a service that provided this capability is the Bitcoin Message Service [6].

3.2 Protocol

In this section, we describe the core protocol for the mixing of a single “chunk” of A 's funds. At a high level, the protocol proceeds as follows: the mix announces mix parameters, the user opts in to the mix, the mix sends a partial warranty back to the user, the user transfers funds, the mix completes the warranty by posting to the public log, the user unblinds the output address, and finally the mix transfers funds to the output address. See Fig. 1 for an illustration of the protocol and to trace the protocol steps. We also indicate the differences in Blindcoin and Mixcoin at each step in the protocol. For ease of presentation, we use the notation $[x]_k$ for a message x committed with commitment function k , and the notation $\{x\}_k$ to represent a signature on message x with signing key k . We also assume that whenever a signature on a message is sent, the entire message x is sent as well in case the contents of the message need to be recovered.

k_{in}	the address from which the A pays, possibly linked to A 's true identity
k_{out}	the address to which the user wishes funds transferred
k_{esc}	an escrow address, unique for each user, that M provides for A to pay
k'_{esc}	an escrow address that M uses to pay to k_{out}
A'	an anonymous identity that A can use to post to the public log
M_{pub}	the public key of M
M_{priv}	the private signing key of M
A_C	a secret commitment/encryption function of A
$A_{C'}$	the inverse of A_C
ω	the number of blocks M requires to confirm A 's payment
n	a per-user nonce, used to determine payment of randomized mixing fees
v	the value (chunk size) to be mixed
ρ	the mixing fee rate A will pay
T	the token, which is the triple (k_{out}, n)
t_1	the time by which A must v BTC to k_{esc} in order to participate in the mix
t_2	the time by which M must post the token T to the public log
t_3	the time by which A' must unblind the output address via the public log
t_4	the time by which the mix must transfer v BTC to k_{out}
D	the mix parameters, a tuple $\{t_1, t_2, t_3, t_4, v, \omega, \rho\}$
Beacon	the beacon function, a publicly verifiable random function

Setup. The mix M publishes a set of *mix parameters*, D , to the public log. This data includes times (t_1, t_2, t_3, t_4) by which different steps of the protocol must be completed; the chunk size, v , which is the amount of BTC that each user inputs into the transaction; ρ , the fraction of user inputs that will be kept as a mixing fee; and ω , the number of blocks that the mix requires to verify the users payments. All of these parameters are part of the original Mixcoin protocol, except for two additional time deadlines.

**Fig. 1.** The Blindcoin protocol

User Sends Offer. See step (1). The user A opts in to the mix by sending its *offer* to M . The offer includes the mix parameters followed by a blinded *token* T . The token consists of the output address and a private, randomly selected nonce n which is used for fee collection. In order to keep these values hidden, the token is encrypted using a commitment function A_C known only to A (who also knows the inverse A_C'). The offer has the following form: $(D, [T]_{A_C})$, where $T = (k_{out}, n)$. In Mixcoin, the output address is not blinded, and the offer has the form: $(v, t_1, t_2, \omega, k_{out}, \rho, n)$.

Mix Sends Partial Warranty. See step (2a). If M accepts the offer, it sends a *partial warranty* back to the user. The partial warranty consists of the blinded token, an escrow address for the user to pay to, and the mix parameters. This is all signed with M_{priv} . The partial warranty has the form $\{[T]_{A_C}, k_{esc}, D\}_{M_{priv}}$. Note that given this partial warranty the user cannot recover the signed token directly, since other fields were included in the signed message. Also, the escrow addresses that the mix provides should be unique to each user, so that the mix can verify which users have paid. In Mixcoin, the mix simply sends back the warranty $(\{v, t_1, t_2, \omega, k_{out}, \rho, n\}_{M_{priv}})$.

Mix Rejects Offer. See step (2b). If the mix rejects the offer, the user destroys the output address. This step is the same as in Mixcoin.

User Pays. See step (3a). A then transfers v coins from any input address k_{in} to k_{esc} by time t_1 . This step is the same as in Mixcoin.

User Fails to Pay. See step (3b). If the user fails to transfer the funds on time, then both parties abort the protocol. This step is the same as in Mixcoin.

Mix Completes Warranty. See step (4a). Once the user has transferred the funds, M must *complete the warranty* by signing the blinded token and publishing it to the public log by time t_2 (which should be long enough after t_1 to allow the transaction to be at least ω blocks deep into the chain). By publishing a user's token, the mix is publicly acknowledging that the user did indeed transfer their funds to the escrow address on time. The fact that this is public allows any third party verifier to check that the mix has completed the warranty by time t_2 . The signed blinded token has the form $\{[T]_{A_C}\}_{M_{priv}}$. Mixcoin does not have an equivalent step, since the user already has the warranty.

Mix Fails to Complete Warranty. User Publishes Incriminating Evidence. See step (4b). If M fails to publish a user A 's token by t_2 , A can publish information to incriminate M . A can present the following evidence: the partial warranty $\{[T]_{A_C}, k_{esc}, D\}_{M_{priv}}$, the transaction $Transfer(v, k_{in}, k_{esc})$ present in the block chain before time t_1 , and the fact that the signed token was not published to the public log before t_2 (this may require an enumeration of all messages in the blockchain between times t_1 and t_2). Any third party verifier can see that M has signed the partial warranty, confirm that someone has transferred funds

to k_{esc} , and verify that indeed no token of the correct form was published to the public log before t_2 , proving that M deviated from the protocol. Mixcoin does not have an equivalent step.

User Anonymously Unblinds Output Address. See step (5). Once their signed blinded token of the form $\{[T]_{A_C}\}_{M_{priv}}$ has been published to the public log, A can apply $A_{C'}$ to recover the signed unblinded token $\{T\}_{M_{priv}} = \{k_{out}, n\}_{M_{priv}}$. The user connects anonymously as user A' and unblinds k_{out} by posting the signed token to the public log. The mix M can verify that the output address is valid, since the signed token is a proof-of-knowledge that the mix signed the corresponding commitment. If A' fails to publish the unblinded token to the public log by time t_3 , M can choose to either refund the coins back to A or retain them. Since A breached the protocol, it cannot produce evidence to incriminate M , so M can do as it pleases with the funds. Mixcoin does not have an equivalent step.

Mix Computes Beacon Function. If the user unblinds their output address by time t_3 , M computes a beacon function $\mathbf{Beacon}(t_3, \omega, n)$ for each (k_{out}, n) pair to determine which output addresses to collect mixing fees from (by not sending any coins to them). The beacon function is a publicly verifiable function that uses entropy collected from the block chain to produce a number uniformly in the range $[0, 1]$ (see [13, 16]). If the value for a particular input is less than or equal to the value ρ from the mix parameters, the chunk destined for that output address is kept by M as a mixing fee. Otherwise, the protocol proceeds to the next step. This step is equivalent to Mixcoin's **Beacon** step, except that in Blindcoin the mix does not know which input addresses correspond to which (k_{out}, n) pairs.

Mix Pays to Output Address. See step (6a). If M acts honestly, then before time t_4 it will transfer v BTC to all unblinded output addresses that have passed the **Beacon** function. The mix does not know which input and output addresses are from the same user, so the mapping from input to output addresses does not matter. Mixcoin has an equivalent step.

Mix Steals Coins. See step (6b). M steals the funds and fails to transfer a chunk to each of the output addresses by time t_4 . Again, Mixcoin has an equivalent step.

User Detects Theft and Publishes Incriminating Evidence. See step (7). The user A detects the theft at time t_4 (since no funds were transferred to its output address), and can publish information to incriminate M . The user publishes the following: the commitment function A_C and its inverse $A_{C'}$, the partial warranty $\{[T]_{A_C}, k_{esc}, D\}_{M_{priv}}$ in the public log, the transaction $\text{Transfer}(v, k_{in}, k_{esc})$ present in the block chain before time t_1 , the signed token $\{k_{out}, n\}_{M_{priv}}$, and the fact that no such transaction $\text{Transfer}(v, k'_{esc}, k_{out})$ is present in the block chain before time t_4 . Any third party can verify that the token was signed with M_{priv} and recover the contents of the signed token with $A_{C'}$. Then, the verifier can check the public log and the block chain to see if both parties followed the protocol by transferring funds and posting to the public log by the

correct deadlines. In Mixcoin, the incriminating evidence that the user publishes consists of the warranty ($\{v, t_1, t_2, \omega, k_{out}, \rho, n\}_{M_{priv}}$), the transaction $Transfer(v, k_{in}, k_{esc})$ present in the blockchain before time t_1 , and the fact that no transaction $Transfer(v, k'_{esc}, k_{out})$ exists in the blockchain before time t_2 .

4 Analysis

In this section, we discuss properties of Blindcoin, evaluate its efficiency and usability, and discuss side channel attacks.

4.1 Properties

The Blindcoin protocol inherits many properties from the Mixcoin system. Below, we outline the properties of both systems and discuss any changes that occur due to our modifications.

Accountability. This property is inherited from Mixcoin. In our modifications, we ensure that this property is preserved.

We define *mix accountability* to be the property that if the mix deviates from the protocol, their cheating can be provably exposed. A mix can deviate from the protocol and steal the user's funds, but the user can then publish their mix-signed warranty along with other evidence to show that the mix has cheated. A mix can steal user funds at several points in the protocol, but at each point the user is “protected” by their warranty.

The warranty is slightly more complex in Blindcoin than in Mixcoin. When the mix agrees to a user's (blinded) offer, they issue a partial warranty back to the user, which is an agreement saying “if the user pays, the mix will publish the signed blinded token to the public log by the warranty deadline.” The mix waits until the user has paid to publish the token to the public log so that it can make sure that only valid, paid-for tokens are in the log. The reason that the token must be published to the log is so that third party verifiers can check that the mix has acknowledged a user's payment. As described in Sect. 3.2, if the mix does not publish the signed blinded token to the public log before the agreed-upon deadline, the user can present a proof of misconduct, which would be damaging to the reputation of the mix.

Another point in time that the mix could potentially cheat is when determining which output addresses to send funds to. If users have correctly followed the protocol and published their signed, unblinded tokens to the public log, then any party can see that the mix has signed the token. Furthermore, after the mix computes **Beacon** to determine the random collection of fees, it is possible for any party to verify that it has correctly computed the function, since **Beacon** is public. If the mix fails to transfer funds to an output address that has passed **Beacon**, then the user can again present a proof of misconduct.

Anonymity. Mixcoin provides strong anonymity guarantees for its users except for the case in which the mix is the adversary. If a mix stores records, they could potentially release them to deanonymize users. In Blindcoin, we extend many of the guarantees provided by Mixcoin and show that anonymity is possible even against a malicious mix. As in the Mixcoin protocol, we assume than an adversary wishes to link an output address k_{out} to the corresponding input address k_{in} .

The first adversarial model we consider is that of a *global passive adversary*. Since the Bitcoin block chain is public and anyone can access all Bitcoin transactions (and the public log in Blindcoin), this is the weakest adversarial model that we consider. Both Mixcoin and Blindcoin provide guarantees that no passive adversary can link input/output address pairs within a particular mix. Thus, Blindcoin achieves k -*anonymity* [33] within the set of all non-malicious users participating in the mix simultaneously. There are no constraints on the number of users that can participate in a mix simultaneously except for the mix server's resources, so the more participants there are, the larger the anonymity set.

A second adversarial model is one of an active attacker who is able to compromise some subset of the input/output address pairs for a particular mixing operation. For both Mixcoin and Blindcoin, the anonymity set for a user remains the number of non-compromised address pairs. An active adversary can also carry out a Sybil attack [19] to convince a user that their anonymity set is larger than it really is. Although there is no real solution to this type of attack, mixing fees make it expensive to carry out.

We also consider the model where the mix is the adversary. This could occur if a mix is compromised or coerced into revealing its records. In the case of Mixcoin, all input/output mappings are revealed, since the mix has access to this information. This problem can be alleviated if the user sends their coins through multiple independent mixes, but a strong adversary could potentially compromise all of them. This also causes a significant degradation in performance and increase in price since user must repeat the entire mixing procedure multiple times. However, a compromised mix does not weaken the anonymity guarantees for Blindcoin since mixes are blinded to the mapping from input to output addresses through the use of the blinded token scheme. This is the main contribution of Blindcoin.

One property that Mixcoin provides is *mix indistinguishability*. This property is unique to Mixcoin, and no other mixing services provide this same guarantee [13]. Against a global passive adversary, this property extends a user's anonymity set to *all* users participating in *different* mixes that use the same parameters for mixing. Unfortunately, this does not hold for Blindcoin, since in the process of unblinding their output addresses, users must publish their signed tokens to the public log. This allows an adversary to link the output addresses to the signing key of the mixing server, defeating mix indistinguishability. We note that mix indistinguishability breaks for Mixcoin when adversaries are active or the mix is malicious. From a user perspective, the loss of mix indistinguishability implies that their anonymity set is limited to only other users participating in the same mix simultaneously (the typical case for Bitcoin mixes), so users must ensure

that this set is large enough for their anonymity purposes. With Blindcoin, a user can easily check the public log *a posteriori* to determine the number of users that participated in their mix.

If enough care is not taken, side channels can leak user information that may lead to deanonymization and address linkage [10, 13, 30]. These side channels include timing, precise values, network layer information, and interactions with the public log.

- *Timing.* To avoid timing analysis attacks, users should not make their interactions with the mix predictable. For example, if a malicious mix posts a particular signed blinded token to the public log and then an output address is unblinded immediately after, the mix may be able to link the token to the output address.
- *Precise Values.* The size of a user’s anonymity set is equal to the number of (non-compromised) users participating in a mixing exchange with the same mix parameters. If each user negotiated its own set of parameters with the mix, then it would be trivial for anyone to link a user’s input and output addresses together. For example, if users had unique unblinding deadlines, then an observer could easily map an unblinded output address back to its associated input address.
- *Network-Layer Information.* For example, two identities connecting with the same IP address could be linked. We assume that users connect via a secure anonymity network such as Tor [18] and keep their connecting identities A and A' separate and unlinkable. In practice, keeping these connecting identities from being linked may be hard against a well-provisioned adversary, but our system does not have a solution to this problem and we consider it outside the scope of this paper.
- *Public Log.* If transactions fees are required to post messages to the public log, users must be careful not to pay the fees from linkable addresses.

Resilience to Denial of Service Attacks. This property is inherited from Mixcoin. In many distributed mixing protocols, a single user can DoS the entire exchange by participating in the protocol up to a certain point, and then refusing to transfer funds, causing the whole operation to fail [24, 32, 34]. However, in both Mixcoin and Blindcoin, each user interacts only with the mixing server, and refusing to comply with the protocol does not affect any other users or slow down the mixing process. One could also attempt to carry out a DoS attack to prevent transactions from entering the blockchain on time or to prevent messages from being posted to the public log. However, these attacks would be difficult to carry out in practice, since they would require the attacker to control a large portion of the Bitcoin block mining pool.

Scalability. This property is inherited from Mixcoin. It is efficient to add more users to a mixing operation since users interact only with the centralized mix and not each other. Further, if having a single server proves to be a bottleneck,

it is possible to load balance the function of the mix onto several different servers, all operating with the same cryptographic keys.

Incentive for Participation. Both users and mixes are incentivized to participate in the system. For a small fee, users are able to safely mix their bitcoins. Although there is some delay associated with a mixing operation in Blindcoin, the fact that one does not need to trust any party clearly makes this a valuable tool. Mix servers are also incentivized. The fees collected by the mix can be set so that they are sufficient to cover operational costs, and make it a worthwhile to provide the service.

Backwards-Compatibility. Blindcoin does not require any changes to the existing Bitcoin protocol, so it can be easily deployed.

4.2 Overheads

A successful Blindcoin mixing operation requires two message directly between A and M , two messages (one from each) posted to the public log, and two Bitcoin transactions. The two messages to the public log are the extra overhead of our protocol over Mixcoin. Each of these new messages come with a deadline by which they must be posted. The time in between deadlines must allow the previous message to be at least ω blocks deep into the block chain to prevent double spending (a typical value might be $\omega = 6$). The expected time between blocks is 10 min, so on average it would take one hour for a message to be 6 blocks deep into the chain. However, the time deadlines should be set much further apart to allow for variations in the time it takes to mine each block. Since the protocol requires four deadlines, and the gap between deadlines is the dominating factor in how long the protocol takes to run, our estimate of the total time for a mixing operation is on the order of a few hours.

Posting messages to the public log (assuming it is implemented in the Bitcoin block chain) costs extra in transaction fees, in addition to the fees paid for the funds transfer. According to [9], the typical transaction fee rate is 0.0001 BTC per 1000 bytes. Although the exact message size depends on the implementation, we believe 5000 bytes is a reasonable estimate, for a total cost per message of 0.0005 BTC per message. Overall, the financial cost to a user is composed of the mixing fee and the transaction fees. For a chunk size of 0.1 BTC, a fee rate of 0.01, and a transaction fee of 0.0005 BTC per message, the total cost to the user is around 0.002 BTC or 2 %, which we believe is a reasonable price to pay for the anonymity benefits.

To summarize, the main downsides of Blindcoin are the loss of mix indistinguishability, the necessity to maintain two unlinkable identities A and A' , and the additional costs and delays associated with posting messages to the public log. The benefits are that the mixing server does not learn the input/output address matching, so multiple rounds of mixing are not required to achieve an adequate level of anonymity.

5 Conclusion

De-anonymization of the Bitcoin block chain is a problem that has attracted much attention. Many different designs of Bitcoin mixing services have been proposed, but we find that each of these systems lacks certain desirable features. In order to make steps towards an ideal system, we propose a list of properties that a mixing system should offer. We believe an ideal mix should be all of the following: accountable, anonymous, scalable, resilient, incentivized, and compatible with the original system.

We present a mixing system that meets the above requirements. The proposed system, Blindcoin, modifies the Mixcoin [13] mixing protocol by using blind signatures [14,23] and a public append-only log. The log makes it possible for a third party to verify the validity of accusations when blind signatures are used. The system retains many of the benefits of the Mixcoin system, while also relaxing the constraint that the mix must be trusted to keep the input/output address mappings of users hidden.

Acknowledgements. We would like to thank the anonymous reviewers, as well as Nadia Heninger, Andrew Miller, Dave Levin, and Bobby Bhattacharjee for their helpful comments and input.

References

1. Bitcoin fog. <http://www.bitcoinfog.com/>
2. Bitlaundry. <http://app.bitlaundry.com/>
3. Bitmixer.io. <https://bitmixer.io/>
4. The current state of coin-mixing services. http://www.thebitcoinreview.com/site.php?site_id=759
5. Online anonymity is not only for trolls and political dissidents. <https://www.eff.org/deeplinks/2013/10/online-anonymity-not-only-trolls-and-political-dissidents>
6. Bitcoin message service, October 2011. <https://bitcointalk.org/index.php?topic=47283.0>
7. Coinswap, October 2013. <https://bitcointalk.org/index.php?topic=321228>
8. blockchain.info, October 2014. <https://blockchain.info/>
9. Transaction fees, March 2014. https://en.bitcoin.it/wiki/Transaction_fees
10. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 34–51. Springer, Heidelberg (2013)
11. 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)
12. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy (SP). IEEE (2014)
13. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: anonymity for bitcoin with accountable mixes. IACR Cryptology ePrint Archive, 2014:77 (2014)

14. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) *Advances in Cryptology*, pp. 199–203. Springer, Heidelberg (1983)
15. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **24**(2), 84–90 (1981)
16. Clark, J., Hengartner, U.: On the use of financial data as a random beacon. *IACR Cryptology ePrint Archive*, 2010:361 (2010)
17. Dingledine, R., Mathewson, N.: Anonymity loves company: usability and the network effect. In: WEIS (2006)
18. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. Technical report, DTIC Document (2004)
19. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) *IPTPS 2002. LNCS*, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
20. Doyle, T., Veranas, J.: Public anonymity and the connected world. *Ethics Inf. Technol.* **16**(3), 207–218 (2014)
21. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) *CRYPTO 2006. LNCS*, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
22. Fleder, M., Kester, M.S., Pillai, S.: Bitcoin transaction graph analysis (2014). <http://people.csail.mit.edu/spillai/data/papers/bitcoin-transaction-graph-analysis.pdf>
23. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. *IACR Cryptology ePrint Archive*, 2009:320 (2009)
24. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world, August 2013. <https://bitcointalk.org/index.php?topic=279249>
25. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 397–411. IEEE (2013)
26. Möser, M., Bohme, R., Breuker, D.: An inquiry into money laundering tools in the bitcoin ecosystem. In: eCrime Researchers Summit (eCRS), pp. 1–14. IEEE (2013)
27. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Consulted **1**(2012), 28 (2008)
28. Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. *Future Internet* **5**(2), 237–250 (2013)
29. Palme, J., Berglund, M.: Anonymity on the internet. Accessed 15 August 2009 (2002)
30. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Altshuler, Y., Elovici, Y., Cremer, A.B., Aharony, N., Pentland, A. (eds.) *Security and Privacy in Social Networks*. Springer, Heidelberg (2013)
31. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) *FC 2013. LNCS*, vol. 7859, pp. 6–24. Springer, Heidelberg (2013)
32. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Practical decentralized coin mixing for bitcoin. HotPETS, Coinshuffle, July 2014
33. Sweeney, L.: k-anonymity: a model for protecting privacy. *Int. J. Uncertainty Fuzziness Knowl.-Based Syst.* **10**(05), 557–570 (2002)
34. Yang, E.Z.: Secure multiparty bitcoin anonymization, July 2013. <http://blog.ezyang.com/2012/07/secure-multiparty-bitcoin-anonymization/>

Privacy-Enhancing Overlays in Bitcoin

Sarah Meiklejohn¹ (✉) and Claudio Orlandi²

¹ University College London, London, UK

s.meiklejohn@ucl.ac.uk

² Aarhus University, Aarhus, Denmark

orlandi@cs.au.dk

Abstract. In this paper, we explore the role of privacy-enhancing overlays in Bitcoin. To examine the effectiveness of different solutions, we first propose a formal definitional framework for virtual currencies and put forth a new notion of anonymity, *taint resistance*, that they can satisfy. We then approach the problem from a theoretical angle, by proposing various solutions to achieve provable taint resistance, and from a practical angle, by examining the taint resistance of the Coinjoin protocol.

1 Introduction

Virtual currencies have existed in various forms—e.g., customer loyalty programs—for decades, yet only in recent years have they exploded in popularity. The initial virtual currency driving this success, Bitcoin, was introduced in January 2009; today, it boasts an exchange rate of over 250 EUR per bitcoin and is in the process of integrating into the traditional banking system via payment gateways such as Bitpay—which as of March 2014 had signed up 26,000 merchants to accept the currency—and partnerships between Bitcoin exchanges such as Bitcoin.de and banks such as Fidor Bank AG. This tremendous growth has impelled regulators and law enforcement officials around the world to take a stand on virtual currencies, with the European Central Bank calling Bitcoin “the most successful—and probably most controversial—virtual currency scheme to date” [7] and the Federal Bureau of Investigation cautioning that, within the Bitcoin network, “law enforcement faces difficulties detecting suspicious activity, identifying users and obtaining transaction records” [8].

This tension between the growing popularity of virtual currencies and their perceived anonymity provides a unique problem for both users of these currencies and for regulators seeking to understand the true risks that they pose. The initial perception of Bitcoin was arguably that it provided anonymity, as evidenced by its adoption in the underground marketplace Silk Road (where bitcoins could be exchanged for goods such as drugs, firearms, and assassins) and by criminals running ransomware such as CryptoLocker or Ponzi schemes [18]. A recent line of research [2, 12, 15, 16, 19], however, showed that it was often possible to trace the movement of bitcoins throughout the network, so as a result the average Bitcoin user was not achieving much anonymity at all.

Perhaps in reaction to these results, a variety of new privacy-enhancing techniques have been proposed for virtual currencies. These techniques can be split into roughly two types: the first type introduces a new virtual currency—such as Zerocash [4, 13] or DarkCoin—that seeks to improve on the anonymity properties of Bitcoin, and the second type proposes *overlays* that can be used without modifying the existing Bitcoin protocol. It is this latter approach that we focus on in this paper.

The main obstacle towards achieving anonymity in Bitcoin is its inherent transparency: while peers can identify themselves using a variety of pseudonyms, every transaction that has ever taken place—and thus the entire spending history of any given bitcoin—is globally visible. One method for improving anonymity in Bitcoin is to *mix* bitcoins together as follows: Alice holds 1 bitcoin and wishes to send it to Charlie, and Bob holds 1 bitcoin and wishes to send it to Dora. If Alice sends her bitcoin to Dora and Bob sends his to Charlie, then they have now essentially swapped the spending histories of these bitcoins; if Alice is a thief, Bob a legitimate user, and Charlie the exchange where Alice wants to cash out her bitcoin, then this swap has effectively “cleaned” the stolen bitcoin. To address the difficulty of finding users to mix with, *mix services* such as Bitcoin Fog and BitLaundry accept bitcoins and—in exchange for a fee—promise to send untainted bitcoins (i.e., bitcoins independent of the ones it received) to any address provided by the user.

Until a year ago, mix services were fairly unattractive, as a fair amount of trust was required to assume that a user would in fact receive his promised bitcoins. In August 2013, however, Gregory Maxwell proposed Coinjoin [10], which promised trustless mixing, and essentially provided a way for users like Alice and Bob in the above example to mix their bitcoins in a single transaction, without relying on either a central service or even any trust in each other. Since then, numerous other trustless mix services have been introduced, such as SharedCoin (sharedcoin.com) and CoinWitness [11], and SharedCoin has been integrated into blockchain.info’s popular wallet service (as of November 2013).

Our Contributions. In this paper, we examine the landscape of privacy-enhancing overlays for Bitcoin in an attempt to determine the extent to which they succeed in achieving anonymity, and the extent to which their anonymity can be strengthened. By looking at the problem from these dual perspectives, we can obtain a picture of both what is feasible in theory and what happens in practice.

To understand the extent to which existing overlays achieve anonymity, it is first necessary to have a common framework in which these techniques can be analyzed. In Sect. 2, we propose such a framework and provide—to the best of our knowledge—the first formal definition of anonymity, *taint resistance*, that can be satisfied by Bitcoin and related virtual currencies. (The main previous definition of anonymity for virtual currencies, unlinkability, cannot be met here.) In Sect. 3, we then analyze how different mechanisms can provide taint resistance. While our framework provides a way to analyze the security of protocols, it is also important to understand the performance of these protocols in practice.

In Sect. 4, we thus perform an experimental analysis of the Coinjoin protocol. We do so from the perspective of both a passive adversary, who uses only the publicly available data from the transaction ledger, and — by engaging in our own coinjoin transactions — an active adversary, who uses its own participation to attempt to de-anonymize the activity of other participants. We find that naïve versions of these adversaries are not particularly successful in identifying which input addresses taint a given output address, and moreover that their success is heavily tied to the quirks of the Coinjoin protocol being used.

2 Definitions and Notation

We begin with formal specifications of decentralized electronic cash (or e-cash) and Coinjoin. We then introduce a notion of anonymity for virtual currencies that we call *taint resistance*.

2.1 Distributed Electronic Cash

To the best of our knowledge, the only existing formal definitions of anonymity in electronic cash are either in a centralized setting or the definition put forth for Zerocoin [13]. The former definitions are out of the scope of this work, as all our currencies of interest are fundamentally decentralized. The Zerocoin definitions do explicitly consider decentralization, but are not applicable in our setting because their definition of anonymity still closely matches the standard cryptographic notion of unlinkability (where, briefly, a user should not be able to distinguish between two coins), which is impossible to achieve for Bitcoin and the many altcoins that it has inspired.

With these considerations in mind, we set out to create a formal definition that encompasses existing virtual currencies. We define a *decentralized e-cash* protocol as a set of PT algorithms (KeyGen , Mint , Spend , Verify , KeyCheck , HistCheck , ValCheck) that behave as follows: via $(pk, sk) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$ one can generate a keypair; via $(hist, \{coin_i, sk_i\}_i) \xleftarrow{\$} \text{Mint}(hist, aux)$ one can generate coins and create a record of this generation; via $tx \xleftarrow{\$} \text{Spend}(hist, \{coin_i, sk_i\}_{i \in [m]}, \{pk_j, v_j\}_{j \in [n]})$ one can send v_j coins to recipients pk_j ; via $\text{accept/reject} \leftarrow \text{Verify}(hist, tx)$ one can (deterministically) decide if a transaction is valid or not; via $\text{KeyCheck}(sk, coin)$ one can (deterministically) determine if a coin is compatible with a given secret key; via $\text{HistCheck}(hist, coin)$ one can (deterministically) determine if a coin is compatible with a given history; and via $\text{ValCheck}(v, \{coin_i\}_i)$ one can (deterministically) determine if a set of coins is compatible with a given value.

To define correctness of the protocol, we say that a set $\{(coin_i, sk_i)\}_i$ is *valid* with respect to a given history $hist$ and value v if (1) $\text{KeyCheck}(sk_i, coin_i) = \text{accept}$ for all i , (2) $\text{HistCheck}(hist, coin_i) = \text{accept}$ for all i , and (3) $\text{ValCheck}(v, \{coin_i\}_i) = \text{accept}$. We say that the protocol achieves correctness if $\text{Verify}(hist', \text{Spend}(hist, \{(coin_i, sk_i)\}_i, \{(pk_j, v_j)\}_j)) = \text{accept}$ for all sets $\{(coin_i, sk_i)\}_i$ that are valid with respect to $hist$ and $\sum_j v_j$, and for all $hist'$ such that $hist \subseteq hist'$.

2.2 Coinjoin

The Coinjoin protocol is designed to make a simplistic taint tracking more difficult. If we consider how bitcoins get spent (or the inputs to the Spend algorithm specified above), we recall that the sender needs to create a transaction using the secret key corresponding to each of the input public keys. One might thus think that a party forming a transaction needs to know each of these keys, so that the input keys in a transaction are all owned by the same entity. Indeed, this observation has formed the basis of a number of heuristics used to cluster together different Bitcoin addresses [2, 12, 15, 16, 19]. In fact, the signatures required in a transaction can be formed in a distributed manner, so that—while highly indicative of single ownership until a year ago—multiple entities can come together to form an ordinary-looking transaction without having to share secret signing keys at all. Coinjoin exploits this ability to distribute the generation of a transaction, and in its simplest form operates as seen in Algorithm 2.1.

Algorithm 2.1. coinjoin: Output a transaction tx

Input: Sets $\{\{\text{coins}_{k,i}\}_{i \in [m_k]}, \{pk_{k,j}, v_{k,j}\}_{j \in [n_k]}\}_{k \in [N]}$ belonging to each of N parties.

- 1 Each party k sends the set $\{pk_{k,j}, v_{k,j}\}_{j \in [n_k]}$ to all the other parties.
 - 2 Define $S_o = \cup_{k \in [N], j \in [n_k]} \{pk_{k,j}, v_{k,j}\}$; after the previous step, each party can now compute S_o . Each party k further computes $\sigma_{k,i} \xleftarrow{\$} \text{Sign}(sk_{k,i}, \text{hist}, S_o)$ for all i , $1 \leq i \leq m_k$.
 - 3 Each party k sends the set $\{pk_{k,i}, \sigma_{k,i}\}_{i \in [m_k]}$ to all the other parties.
 - 4 Define $S_i = \cup_{k \in [N], i \in [m_k]} \{pk_{k,i}, \sigma_{k,i}\}$; after the previous step, each party can now compute S_i . The final transaction is $\text{tx} = (S_i, S_o)$, which each party can broadcast to the Bitcoin network.
-

Briefly, each party broadcasts their desired output keys to the other parties, individually signs the completed list, and then broadcasts the signatures to the other parties; each party can then broadcast this list of signatures and recipients—presumably in randomized order to prevent trivial de-anonymization—as the collective transaction. The way in which the order and the transaction fees are determined can be used to identify the behavior of different Coinjoin services, as we will see in Sect. 4.

To attempt to understand the growing popularity of coinjoin transactions, we looked at how many transactions matched a rough pattern of what we might expect coinjoins¹ to look like. We defined this pattern as any transaction having more than five inputs and more than five outputs, and looked at how many transactions matched this pattern (which is admittedly only roughly correlated with coinjoins, but as coinjoins are indistinguishable from other transactions

¹ In the rest of the paper, we follow the convention of called the protocol Coinjoin and the resulting transactions coinjoins.

we cannot do better than an approximation). Prior to August 2013, we saw an average of zero matching transactions per block, but afterwards we saw a steady increase: first to an average of two matching transactions per block in December 2013, then to a peak of 14 in March 2014, and finally settling to an average of 10 in July 2014, where it has remained ever since.

2.3 Taint Resistance

As mentioned above, the existing notions of unlinkability for electronic cash require that a valid coin belonging to one user is indistinguishable from a valid coin belonging to another. In Bitcoin, it is impossible to satisfy this definition: a bitcoin essentially *is* its spending history, and it is thus trivial to distinguish two valid bitcoins. Any notion of anonymity that is useful for Bitcoin must therefore focus less on the coins themselves and more on ownership. Looking back at the Coinjoin protocol in Sect. 2.2, we can see that it is attempting to obscure not the individual origins of the bitcoins involved—as, again, this can be trivially discerned using the public ledger—but rather the ownership of the bitcoins at each point in their spending histories.

To focus more on ownership, we thus present a notion of anonymity called *taint resistance*, which attempts to capture how well an adversary can discern the ownership of a bitcoin based on its previous spending history. Our definition has the advantage that we can not only provide proofs of security (i.e., prove that a protocol achieves optimal taint resistance), but that it also provides a concrete measurement of the degree to which a proposed solution such as Coinjoin is effective in improving anonymity.

In order to define taint resistance, we must first define what it means for bitcoins to be tainted at all. The naïve version of the definition is simple: if a public key has received some bitcoins as the result of a transaction, then all the inputs to that transaction have tainted those bitcoins (or, in a public-key-based definition, have tainted that output public key). As this is exactly the type of taint analysis that protocols like Coinjoin are designed to thwart, however, we consider a more specific definition.

Definition 2.1 (Taint Set). *For a coinjoin tx produced as specified in Algorithm 2.1 and a public key $pk \in \text{outputs}(\text{tx})$, the taint set for pk is the set $T = \{pk_i\}_{i \in [m]}$ such that $v_{i,j} \neq 0$.*

The definition of the taint set only makes sense when applied to “real” coinjoins, since it explicitly mentions the values $v_{i,j}$ transferred from U_i to U_j . In general, it is impossible to define a taint set unless we make some assumption on how the transaction was generated; to see why, recall the example in the introduction, where Alice and Bob pay 1 coin each to Charlie and Dora. If one looks only at the transaction, there are potentially exponentially many explanations for the transaction; i.e., for each value $v < 1$, Alice and Bob could have been paying $(v, 1 - v)$ respectively to Charlie and $(1 - v, v)$ to Dora (or vice versa).

To consider the success of an adversary, we use the Matthews correlation coefficient (MCC), which is a way to measure the quality of a binary classifier.

This fits well with an adversary attempting to de-anonymize transactions, as we can view it as assigning a ‘yes’ value to an input if it thinks it taints a given output, and a ‘no’ value if it does not. The MCC ranges from -1 to $+1$: -1 indicates a complete mismatch between the underlying truth and the classification, 0 indicates that the classifier does no better than random, and $+1$ indicates a perfect match. (We define $0/0 = 0$.)

For our purposes, we can define all the relevant terms in the original MCC formula using the set A output by an adversary, the underlying taint set T , and the full set of input keys S . Plugging these in, we get the following definition:

Definition 2.2 (Accuracy). *For a transaction tx , a public key $pk \in \text{outputs}(\text{tx})$ and its corresponding taint set T , auxiliary information $aux \in \{0, 1\}^*$, and an adversary \mathcal{A} , we say that the adversary achieves ϵ -accuracy with respect to aux , where*

$$\epsilon = \frac{|A \cap T| \cdot |S \setminus (A \cup T)| - |A \setminus T| \cdot |T \setminus A|}{\sqrt{|A| \cdot |T| \cdot |S \setminus T| \cdot |S \setminus A|}}$$

for $S \leftarrow \text{inputs}(\text{tx})$ and $A \xleftarrow{\$} \mathcal{A}(\text{tx}, aux)$.

Definition 2.3 (Taint Resistance). *For a transaction tx , a public key $pk \in \text{outputs}(\text{tx})$, auxiliary information $aux \in \{0, 1\}^*$, and a set of public keys $S \subseteq \text{inputs}(\text{tx})$, we say that tx achieves ϵ -taint resistance with respect to aux if no (potentially unbounded) adversary \mathcal{A} can achieve more than $(1 - \epsilon)$ -accuracy.*

Unlike accuracy, taint resistance is quantified over all possible adversaries (it is a property of the transaction, not the adversary), and taint resistance ranges from 0 to 1 since there is always a trivial adversary that outputs S or the empty set and thus achieves accuracy 0 . In the sequel, we first identify constructive solutions for achieving taint resistance, and then attempt to analyze the effect that the auxiliary information aux can have on a transaction’s taint resistance.

3 Achieving Taint Resistance

In this section, we describe and analyze different mechanisms for achieving taint resistance. All of the solutions use well-known methods from the cryptographic literature; i.e., we do not claim any novelty in the cryptographic tools, but instead seek to explore how they can be used to achieve taint resistance.

It is instructive to start our discussion by describing how an adversary can identify the taint set of a given public key in a coinjoin; this will also be useful in our experimental analysis in Sect. 4. We describe a few scenarios below:

- **The Transaction Itself.** A coinjoin has two input values x and y and two output values $x - F_x$ and $y - F_y$ (where the transaction fee is $F_x + F_y$). If $x \neq y$ and F_x and F_y are small, the adversary can use this information to increase his belief in which input address transferred value to a given output address.

- **Participating in Coinjoins.** An unlucky user performs a coinjoin with the adversary as his sole mixing partner. Regardless of how the coinjoin is performed, the adversary—who controls one input address and one output address—can trivially compute the taint set of the other output address (assuming one input and one output address per participant).
- **Influencing the Creation of Coinjoins.** An adversary can learn information about the taint set of an output address in a coinjoin by maliciously influencing the way in which a coinjoin is created. This includes not only the protocol for generating the coinjoin, but also the process of choosing the other participants (where, for example, the adversary might try to force an honest user to perform coinjoins with adversarially controlled addresses).

3.1 Using a Trusted Server

We describe now an ideal setting for performing coinjoins; i.e., one that leads to optimal taint resistance. We then proceed to replace some of the (unrealistic) assumptions using cryptographic tools.

To start, we assume a central trusted server, which partitions users into different coinjoins, permutes their addresses at random, and transfers an amount equal to the *minimum* of all input values (and returns the differences to change addresses).

Grouping Users: A user U^j who wish to perform a coinjoin sends his addresses (pk_i^j, pk_o^j, pk_c^j) (for *input address*, *output address* and *change address* respectively) to a central server S , where pk_o^j and pk_c^j are freshly generated addresses. The server S randomly assigns the user to a bucket ℓ of size n .

Performing Transactions: Let U^1, \dots, U^n be the users assigned to a certain bucket ℓ . Let v_i^j be the values associated to pk_i^j and $v^* = \min(v_i^j)$. The server S prepares a transaction with input addresses (pk_i^1, \dots, pk_i^n) and outputs addresses $(pk_o^{\pi(1)}, pk_o^{\pi(n)}, pk_c^1, \dots, pk_c^n)$, for a random permutation π . Let F be the transaction fee: then for each j , the address pk_c^j receives $v_c^j = v_i^j - v^* - F/n$ coins, and all pk_o^j receive the same value v^* .

Signing Transactions: The server S sends the transactions tx to all users U^j in the bucket ℓ . If U^j 's output address is present in the transaction and the amount received by the change address of U^j is correct, U^j signs the transaction and sends it back to the server.

Claim 1. *Any tx generated using the above protocol achieves an expected 1-taint resistance for the first n output addresses, against every adversary who does not control (U^1, \dots, U^n, S) .*

The proof of the claim is trivial: the first n output addresses are fresh unused addresses in a randomly permuted order, are generated independently, and all receive the same amount of bitcoins. Since the adversary does not control any other user participating in the transaction nor the server, the adversary has no auxiliary information about the permutation π .

3.2 Reducing Trust in the Central Server

The solution presented in the previous section offers taint resistance against only simplistic adversaries. We propose here a solution that also works against a partially corrupted server. The solution uses anonymous channels in a crucial way and is a simple application of the results of [9] in this setting.

Grouping Users: User U^j sends his input and change address (pk_i^j, pk_c^j) to the server S , which replies with the index ℓ of the bucket to which the user has been assigned. Now, *using a (different) anonymous channel*,² user U^j sends the pair (ℓ, pk_o^j) to the server.

Performing Transactions: As before.

Signing Transaction: As before.

Intuitively, since the users communicate the input and output key to the server using two distinct anonymous channels, the server cannot link those addresses together.

Claim 2. *Any tx generated using the above protocol achieves an expected $(1 - \epsilon)$ -taint resistance for the first n output addresses, against every adversary who does not control (U^1, \dots, U^n) and corrupts S in a passive way.*

Since S is passively corrupted, we need to add S 's view to the auxiliary information aux given to the adversary. The main idea is that, since each user sends its output public key using a fresh anonymous channel, even the central server does not learn the mapping between the input keys and output keys.

More formally, as shown by [9], our use of the anonymous channel is a secure implementation of an ideal functionality that performs a secure shuffle. Therefore, the view of the server can be efficiently simulated, which implies that any adversary that can achieve non-negligible accuracy in this protocol can be used to do so in the previous protocol as well.

3.3 Removing the Central Server

Finally, we sketch a solution that does not require any central server. Here we think of a high number of users N who want to perform coinjoins and have access to some broadcast channel. It is at this point (conceptually) trivial to let all parties run a secure computation protocol to replace the central server; in practice, however, this is quite cumbersome, as it requires high communication and computational resources.

Instead, we seek a solution that allows the users to partition themselves in smaller sets of (expected) size n and then perform simple “mix-nets” between them (a very similar solution has also been described in [17]). Here it is crucial that users are assigned to groups at random (even in the presence of other actively corrupted users) to guarantee that an adversary who controls few (say $n - 1$) parties cannot force an honest user to perform a coinjoin with those addresses. Let $H : \{0, 1\}^* \rightarrow [n]$ be a cryptographic hash function.

² This can implemented by creating a new identity on Tor.

Grouping Users: All users perform a *simultaneous exchange* of their input and change addresses (pk_i^j, pk_c^j) . (This can be implemented by having all parties commit to their addresses, and then send the opening only after all commitments have been received). Each user j is assigned to group $H(pk_i^j)$.

Performing Transactions: Let U^1, \dots, U^n be the users assigned to a certain bucket ℓ . Now these users can perform a simple mix-nets as follows: User U^1 encrypts his output address pk_o^j under all the public keys of the other parties—i.e., forms $C_2^1 = E_{pk_2}(E_{pk_3}(\dots(E_{pk_n}(pk_o^j))))$ —and sends it to U^2 . Now U^2 samples a random permutation π of $\{1, 2\}$, decrypts the received ciphertexts and computes $C_3^{\pi(1)} = D_{sk_2}(C_2^1)$ as well as $C_3^{\pi(2)} = E_{pk_3}(E_{pk_4}(\dots(E_{pk_n}(pk_o^j))))$ and so on. That is, at every step user U^j removes one layer of encryption from all the ciphertexts he receives, encrypts his own output public key under the public keys of the next users, shuffles the ciphertexts and sends them on to the next user. Finally, U^n decrypts and obtains the full list of the output addresses, and prepares a transaction tx in the same way as the server did in the previous solutions.

Signing Transaction: The user U^n sends the transactions tx to all users U^j in the bucket ℓ . If pk_o^j is present in the transaction and the amount received by pk_c^j is correct, U^j signs the transaction and sends it back to U^n .

Claim 3. *Any tx generated using the above protocol achieves an expected $(1 - n\tau^{n-1})$ -taint resistance for the first n output addresses, against every adversary who controls a fraction τ of parties.*

There are two key ideas here. First, there is no guarantee that a user will perform the correct decryption and shuffle. Still, it suffices to use a passively secure mix-nets protocol, as users will sign the transaction only if their address is present in the final transaction. Second, unless the adversary controls exactly $n - 1$ users in a given bucket, then the resulting mix between the two honest user will leave the adversary without any information to do any better than guessing. Since the adversary cannot control how the users are assigned into buckets (due to the use of commitments and the hash function), assuming that the adversary controls less than $\tau \cdot N$ parties then the probability that he controls exactly the other $n - 1$ users in the bucket assigned to the target public key is less than $n \cdot \tau^{n-1}$.

4 Experimental Analysis

In the previous section, we saw methods for achieving taint resistance. All of these approaches, however, achieved security only with respect to the auxiliary information that an adversary could obtain. In this section, we explore this auxiliary information and consider how much it could affect taint resistance.

We consider different forms of auxiliary information from two perspectives. First, we consider a minimal *passive* adversary that can glean information about coinjoins based only on what it sees in the block chain. To emulate such an adversary, we need only download the Bitcoin block chain, which we did as recently as 2 September 2014 (at which point there were 318,768 blocks, 45.84 million transactions, and 45.79 million distinct public keys).

Next, we consider an *active* adversary that participates in coinjoins. To emulate this adversary, we used `blockchain.info`'s SharedCoin service to participate in our own coinjoins. We used this service 54 times between 30 June 2014 and 11 August 2014; in each transaction, we sent 0.02 BTC from a single address we owned to a new one (freshly generated). As `blockchain.info` requires at least two repetitions of the Coinjoin protocol (in their own words, “a higher number of repetitions makes the transaction more difficult to trace and improves privacy”), this resulted in two transactions: in the first 0.0205 BTC was sent to a freshly generated intermediate address and the remainder was sent to a freshly generated change address, and in the second 0.02 BTC was sent to the address we specified. We therefore ended up with 108 distinct coinjoins, each of which took between 8 and 74 seconds to complete (on average, 30), and had between 4 and 40 input addresses (on average, 14.5) and between 4 and 42 output addresses (on average, 25.8).

4.1 Auxiliary Information Based on Value

One bitcoin is divisible down to the eighth decimal place, meaning that it is often possible to end up with “jagged” bitcoin values. If each user in a coinjoin sends different jagged values, then—as discussed in Sect. 3—these values might help an adversary discover the permutation between input and output addresses. This is an acknowledged limitation of the Coinjoin protocol, and the only solution for achieving full taint resistance is to ensure that all participants send the same amount of bitcoins. From a usability perspective, however, this is not particularly desirable, so it is useful to understand the degree to which differing amounts really degrade taint resistance.

We first consider the different behaviors in which coinjoin users might engage. For example, a user might *aggregate* bitcoins by combining the balances of m separate addresses into one address, and a user might *split* bitcoins by moving the balance of one address into n separate addresses. (This latter case often arises in the case of making change, where $n = 2$.) We attempt to correlate the values in a coinjoin by identifying subsets of the input values whose sum adds up to an output value (the m -to-1 scenario) and which output values are part of a subset whose sum adds up to an input value (the 1-to- n scenario). As these are the only two behaviors that we engaged in our own coinjoins (and because the subset sum problem is **NP**-complete), we do not consider the more general m -to- n setting.

Finally, because transactions have fees, we perform a *noisy* subset sum by allowing the sum of the subsets to potentially exceed the target value by at most the transaction fee. We present our algorithm in Algorithm 4.1.

As a sanity check, we ran this algorithm using the ground truth data collected from our coinjoins; i.e., for each of our known output addresses. (As we knew that we engaged only in 1-to-2 or 1-to-1 transactions, we also “cheated” by considering only subsets of size at most 2.) We then compared this to the known taint set to get the accuracy (see Definition 2.2) of an adversary running this algorithm, and plotted the results in Fig. 1.

Algorithm 4.1. `find_taint`: Output the taint set for a public key

Input: a transaction tx and a public key $pk \in \text{outputs}(\text{tx})$.

- 1 Compute the fee F for tx
- 2 Compute $S_{m\text{-to-}1} \leftarrow \text{noisy_subset_sum}(\text{inputs}(\text{tx}), \text{val}(pk), F)$
- 3 $S_{1\text{-to-}n} \leftarrow \emptyset$
- 4 **forall** the $pk' \in \text{inputs}(\text{tx})$ **do**
- 5 Compute $S \leftarrow \text{noisy_subset_sum}(\text{outputs}(\text{tx}), \text{val}(pk'), F)$
- 6 **if** $pk \in S$ **then**
- 7 $S_{1\text{-to-}n} \leftarrow S_{1\text{-to-}n} \cup \{pk'\}$
- 8 **return** $S_{m\text{-to-}1} \cup S_{1\text{-to-}n}$

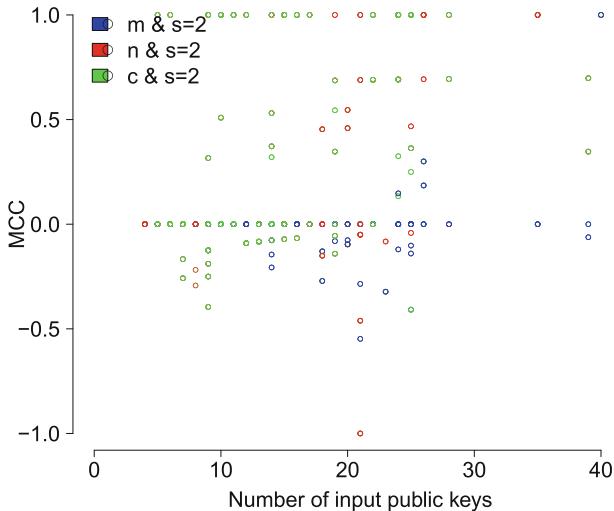


Fig. 1. For our known output addresses, the MCC for Algorithm 4.1 when run on coinjoins with differing numbers of input addresses. The points in blue represent the MCC when considering just m -to-1 taint ($S_{m\text{-to-}1}$), the points in red 1-to- n taint ($S_{1\text{-to-}n}$), and the points in green the combined taint set S . All use a maximum subset size of 2 when running `noisy_subset_sum`.

As we can see, the algorithm was fairly inconsistent in its accuracy. This is due in part to a quirk of `blockchain.info`'s service: in the first step, 0.0205 BTC was sent to an intermediate address, but in the second step, only 0.02 BTC was sent to the final address; the remainder was kept by `blockchain.info` as a service charge. As a result, our algorithm did not positively associate the input and output addresses, so any non-empty set it output was a false positive (as indicated by the largely non-positive MCC for the m -to-1 taint). For the first type of transaction, in which the initial address split its value between the change and intermediate addresses, we see that (unsurprisingly) the 1-to- n and combined taint did fairly well overall, and—especially for transactions with fewer input keys and thus fewer options—often found the taint set perfectly (as indicated by the points with an MCC of 1).

Active Adversaries. To emulate the benefit that an active adversary has—that in a coinjoin in which it participated, it can rule out its own addresses—we started with our own coinjoins, eliminated our own addresses, and ran Algorithm 4.1 on the remainder. As we could no longer use ground truth data to compute the accuracy, we instead plotted the size of the set S . The results are in Fig. 2.

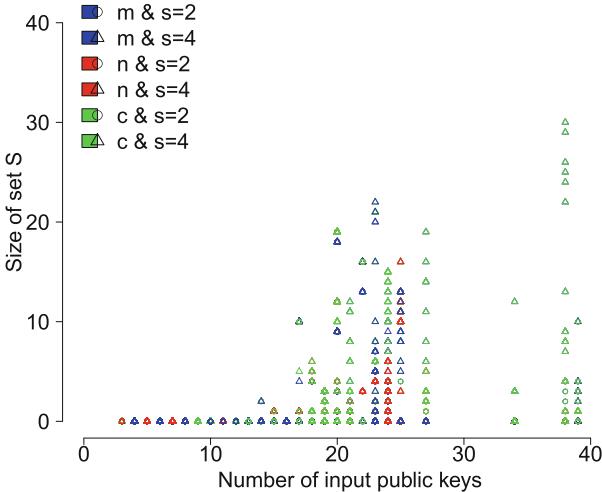


Fig. 2. For the unknown addresses in known coinjoins, the size of the set output by Algorithm 4.1. The points in blue consider just m -to-1 taint ($S_{m\text{-to-}1}$), the points in red 1-to- n taint ($S_{1\text{-to-}n}$), and the points in green the combined taint set S . The value of s is the maximum subset size when running `noisy_subset_sum`.

On average, we can see that the set S was of a fairly small size; in the many cases that it was 0, either (1) a more general m -to- n Coinjoin was used, (2) the input subset was of size larger than 4, or (3) most likely, our basic algorithm did not take into account some quirk of the `blockchain.info` service (as was the case with our own transactions). In the cases where S was larger, we ultimately do not know whether it truly captured the taint set or whether it was simply capturing “noise” from other inputs that happened to sum to the target value. To nevertheless attempt to capture some of this noise, we considered for the case of m -to-1 taint not the possible taint set but rather how many subsets of input addresses summed to the correct output value. Here, a lower value indicates that the algorithm is more sure about which input addresses taint the given output (as there are fewer options), while a larger number of subsets indicates noise from the inputs (for example, if many of the inputs have the same value), and consequently less certainty. Of our 5156 data points, only 62 had exactly one matching subset (again, using only the m -to-1 taint), and 4812 had no matching subsets at all. Our simple active adversary was thus not particularly successful at identifying the taint set, although it did at least manage to avoid false positives.

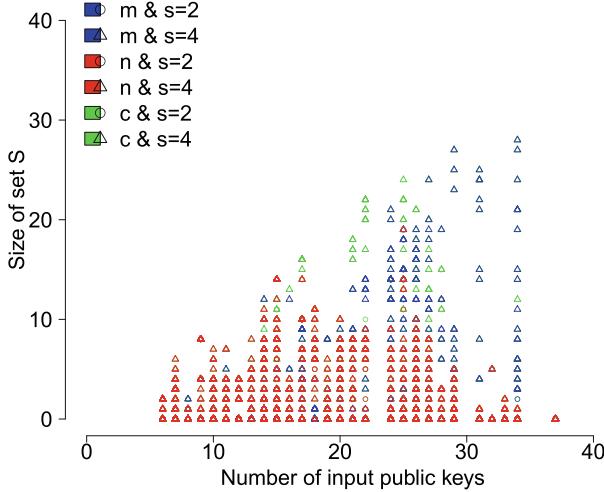


Fig. 3. For potential coinjoins, the size of the set output by Algorithm 4.1. The points in blue consider just m -to-1 taint ($S_{m\text{-}1}$), the points in red 1-to- n taint ($S_{1\text{-}n}$), and the points in green the combined taint set S . The value of s is the maximum subset size when running `noisy_subset_sum`.

Passive Adversaries. Finally, we consider a passive adversary; i.e., one that does not participate in any coinjoins, but instead tries to infer from an examination of the block chain which transactions are coinjoins and which coinjoin input addresses taint which output addresses. To do this, we considered the same pattern used in Sect. 2.2: a potential coinjoin has more than five inputs, more than five outputs, and took place after 30 August 2013. This is of course a very rough heuristic, and before using this data we eliminated any obvious mismatches; this included behavior such as a user combining funds to place bets on different odds in dice and other gambling games. From this set, we then randomly sampled 100 transactions (to match the size of our set of real coinjoins). We again ran Algorithm 4.1 and recorded the size of the set S ; the results are in Fig. 3.

Again, we ultimately cannot know how successfully the analysis identified the taint set. We can, however, see a clear increase in the size of the set S for these passively identified coinjoins over the size for the definite coinjoins. Running the same analysis to determine the number of possible subsets in the m -to-1 scenario, we found that of our 3476 data points, 278 had exactly one matching subset and 2513 had no matching subsets. The subsets produced by the algorithm were thus far noisier than the ones produced for the definite coinjoins, indicating that an active adversary has a distinct advantage over a passive one.

5 Related Work

We consider related work that both proposes ways to enhance privacy in virtual currencies and attempts to analyze the existing anonymity in Bitcoin.

In terms of the latter, perhaps the work most similar to our own is Coinjoin Sudoku [3], in which Atlas analyzes the same mixing service (namely, the SharedCoin one provided by `blockchain.info`) and finds that it “offers only limited privacy to users due to weaknesses in its design.” As we did, Atlas performed his own coinjoins and examined possible relationships between input and output addresses based on the values. He claims to be able to taint the majority of inputs and outputs, but the details of the analysis are not given and as of this writing no source code or tool is available, despite a promised release date of 23 June 2014. A recent line of research has examined anonymity in the overall Bitcoin network [2, 12, 15, 16, 19], and a recent paper by Möser et al. found that centralized mix services do make tracing transactions significantly more difficult [14].

In terms of privacy enhancements, alternative virtual currencies have been proposed such as Zerocash [4, 13] and Darkcoin [1]. There are also a number of proposed overlays for Bitcoin, such as Pinocchio Coin [6], Mixcoin [5], CoinWitness [11], and CoinShuffle [17]. None of these papers formally prove the security of their proposed constructions, but our decentralized construction is almost identical to the one in CoinShuffle and our centralized construction is related to the one in Mixcoin.

6 Conclusions and Open Problems

In this paper, we presented a definitional framework for the anonymity that virtual currencies such as Bitcoin provide. We then provided constructive solutions for achieving this new notion of anonymity, and analyzed the extent to which it was already being achieved by existing Bitcoin overlays. For both of our results, several interesting open problems and extensions remain. Our constructions require additional cryptographic techniques, and it is important to understand the overhead that these techniques require. Similarly, our analysis of SharedCoin was relatively simplistic and did not take into account certain quirks of the service being used. Extending the analysis to consider these quirks—or understanding the extent to which doing so would affect scalability—would provide a more complete picture of the successes and limitations of the Coinjoin protocol.

Acknowledgments. The second author was supported by the Danish National Research Foundation (under the grant 61061130540, CTIC research center) and by the Danish Strategic Research Council (CFEM research center) within which this work was performed.

References

1. Darkcoin. <https://www.darkcoin.io>
2. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 34–51. Springer, Heidelberg (2013)

3. Atlas, K.: Coinjoin Sudoku. <http://www.coinjoinsudoku.com>
4. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: decentralized anonymous payments from bitcoin. In: Proceedings of the IEEE Symposium on Security and Privacy (2014)
5. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: anonymity for bitcoin with accountable mixes. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 486–504. Springer, Heidelberg (2014)
6. Danezis, G., Fournet, C., Kohlweiss, M., Parno, B.: Pinocchio coin: building zero-coin from a succinct pairing-based proof system. In: Proceedings of PETShop 2013 (2013)
7. European Central Bank. Virtual Currency Schemes. ECB Report, October 2012. www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemes201210en.pdf
8. Federal Bureau of Investigation. (U) Bitcoin Virtual Currency Unique Features Present Distinct Challenges for Deterring Illicit Activity. Intelligence Assessment, Cyber Intelligence and Criminal Intelligence Section, April 2012. <http://cryptome.org/2012/05/fbi-bitcoin.pdf>
9. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography from anonymity. In: Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), Berkeley, California, USA, pp. 239–248, 21–24 October 2006
10. Maxwell, G.: CoinJoin: Bitcoin privacy for the real world. Post on bitcointalk.org. <https://bitcointalk.org/index.php?topic=279249>
11. Maxwell, G.: Really Really ultimate blockchain compression: CoinWitness. Post on bitcointalk.org. <https://bitcointalk.org/index.php?topic=277389>
12. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of IMC 2013 (2013)
13. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoins: anonymous distributed E-Cash from bitcoin. In: Proceedings of the IEEE Symposium on Security and Privacy (2013)
14. Möser, M.: An inquiry into money laundering tools in the bitcoin ecosystem. In: Proceedings of the IEEE 2013 eCrime Researchers Summit (2013)
15. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Altshuler, Y., Elovici, Y., Cremers, A.B., Aharony, N., Pentland, A. (eds.) Security and Privacy in Social Networks, pp. 197–223. Springer, New York (2013)
16. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 6–24. Springer, Heidelberg (2013)
17. Ruffing, T., Moreno-Sánchez, P., Kate, A.: Coinshuffle: practical decentralized coin mixing for bitcoin. In: Kutylowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 345–364. Springer, Heidelberg (2014)
18. Securities and Exchange Commission. SEC Charges Texas Man With Running Bitcoin-Denominated Ponzi Scheme, July 2013. www.sec.gov/News/PressRelease/Detail/PressRelease/1370539730583
19. Spagnuolo, M., Maggi, F., Zanero, S.: BitIodine: extracting intelligence from the bitcoin network. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 457–468. Springer, Heidelberg (2014)

Search-and-Compute on Encrypted Data

Jung Hee Cheon¹(✉), Miran Kim¹, and Myungsun Kim²

¹ Department of Mathematical Sciences, Seoul National University,
Seoul, South Korea

{jhcheon,alfks500}@snu.ac.kr

² Department of Information Security,
The University of Suwon, Hwaseong, South Korea
msunkim@suwon.ac.kr

Abstract. Private query processing on encrypted databases allows users to obtain data from encrypted databases in such a way that the user’s sensitive data will be protected from exposure. Given an encrypted database, the users typically submit queries similar to the following examples:

- How many employees in an organization make over \$100,000?
- What is the average age of factory workers suffering from leukemia?

Answering the above questions requires one to **search** and then **compute** over the encrypted databases *in sequence*. In the case of privately processing queries with only one of these operations, many efficient solutions have been developed using a special-purpose encryption scheme (e.g., searchable encryption). In this paper, we are interested in efficiently processing queries that need to perform both operations on *fully* encrypted databases. One immediate solution is to use several special-purpose encryption schemes at the same time, but this approach is associated with a high computational cost for maintaining multiple encryption contexts. The other solution is to use a privacy homomorphic scheme. However, no secure solutions have been developed that meet the efficiency requirements.

In this work, we construct a unified framework so as to efficiently and privately process queries with “search” and “compute” operations. To this end, the first part of our work involves devising some underlying circuits as primitives for queries on encrypted data. Second, we apply two optimization techniques to improve the efficiency of the circuit primitives. One technique is to exploit SIMD techniques to accelerate their basic operations. In contrast to general SIMD approaches, our SIMD implementation can be applied even when one basic operation is executed. The other technique is to take a large integer ring (e.g., \mathbb{Z}_{2^t}) as a message space instead of a binary field. Even for an integer of k bits with $k > t$, addition can be performed with degree 1 circuits with lazy carry operations. Finally, we present various experiments by varying the parameters, such as the query type and the number of tuples.

Keywords: Encrypted databases · Private query processing · Homomorphic encryption

1 Introduction

Privacy homomorphism is an important notion for encrypting clear data while allowing one to carry out operations on encrypted data without decryption. The concept was first introduced by Rivest et al. [17], and much later, Feigenbaum and Merritt's question [12] affirmed the concept: Is there an encryption function $E(\cdot)$ such that both $E(x + y)$ and $E(x \cdot y)$ are easy to compute from $E(x)$ and $E(y)$? Since then, there had been very little progress made in determining whether such efficient and secure encryption schemes exist until 2009, when Gentry constructed such an encryption scheme [13].

While the use of Gentry's scheme and other HE schemes (e.g., [6, 7, 22]) allows us to securely evaluate any function in a theoretical sense, the evaluation cost is still far from being practical for many functions. Among the important functions, we restrict our interest to a set of functions for databases, which raises the following question: *Given a set of fully encrypted databases, can we construct a set of efficient functions to process queries over the encrypted databases?* If so, *what is the computational cost of the functions?*

Although this question is the starting point of this work, to facilitate a better understanding of the approach, we describe the motivation for our work from a different perspective. Currently, perhaps the simplest way to search for records satisfying a particular condition over encrypted databases is via searchable encryption (e.g., [2, 3, 10, 21]). However, privately processing `sum` and `avg` aggregation queries in the same condition is performed using homomorphic encryption (e.g., [5, 11] and [16]). Thus, the private processing of a query that includes both matching conditions and aggregate operations requires the use of two distinct encryption techniques in parallel, *i.e.*, searchable encryption and homomorphic encryption.

Recently, Ada Popa et al.'s CryptDB [1] processed general types of database queries using layers of different encryption schemes: deterministic encryption for equality condition queries, order-preserving encryption for range queries, and homomorphic encryption for aggregate queries. The disadvantage of their work is that in the long run, it downgrades to the lowest level of data privacy provided by the weakest encryption scheme. This observation leads to the natural question: *Can we construct a solution to efficiently address such a database query without maintaining multiple contexts of encryption?* However, there exists no solutions for expressing and processing various queries on *fully* encrypted databases in an *efficient* way.

1.1 Our Results

Our main results are as follows:

- **A unified framework for private query processing:** We provide a common platform so that database users may work on a *single underlying cryptosystem*, represent their query as a function in a conceptually simpler manner, and efficiently carry out the function on fully encrypted databases.
- **Optimizing Circuits and their Applications to Compact Expressions of Queries:** The foundation of our simple framework is a set of optimized

circuits: equality, greater-than comparison and integer addition. We call these *circuit primitives*. Our optimizations of circuit primitives have been taken in such a way as to minimize the circuit depth and the number of homomorphic operations. To do this, we make extensive use of single-instruction-multiple-data (SIMD) techniques to move data across plaintext slots. In general, SIMD technology allows for basic operations to be performed on several data elements in parallel. On the contrary, our proposal works on packed ciphertexts of several data elements and thus enables one to improve the efficiency of the basic operations of circuit primitives. Furthermore, we find that all circuit primitives have $\mathcal{O}(\log \mu)$ depth for μ -bit data.

We then express more complicated queries by a composition of the optimized circuit primitives. The resulting query functions are conceptually simpler than other representations of database queries and are compact in the sense that retrieval queries require at most $\mathcal{O}(\log \mu)$ depth.

- **Further Improvements in the Performance of Query Processing:** HE schemes usually use \mathbb{Z}_2 as a message space so that their encryption algorithm encrypts each bit of message. While our circuit primitives efficiently work on bit encryptions, we can achieve further improvements by adopting a large integer ring (*e.g.*, \mathbb{Z}_{2^t}), especially in the case of *computing* on encrypted numeric-data. Even for an integer of k bits with $k > t$, addition can be performed with degree 1 circuits by processing lazy carry operations. Although this rectification requires to amend our circuit primitives, we can again preserve their optimality by SIMD operations. In other words, search-and-compute queries can be processed with only $\mathcal{O}(\log \mu)$ -depth circuits.
- **Comprehensive Experiments:** We conduct comprehensive experiments for evaluating the performance of various queries expressed by our techniques from a theoretical as well as practical perspective.

1.2 A High-Level Overview of Our Approach

Assuming a database consisting of N blocks, *i.e.*, $R_1 \parallel R_2 \parallel \dots \parallel R_N$, to encrypt the record R_i , a DB user prepares a pair of public/private keys (pk, sk) for a HE scheme and publishes the public key to a DB server. The users store their encrypted records $\bar{R}_i = E_{pk}(R_i)$ for $1 \leq i \leq N$ in the same way as normal

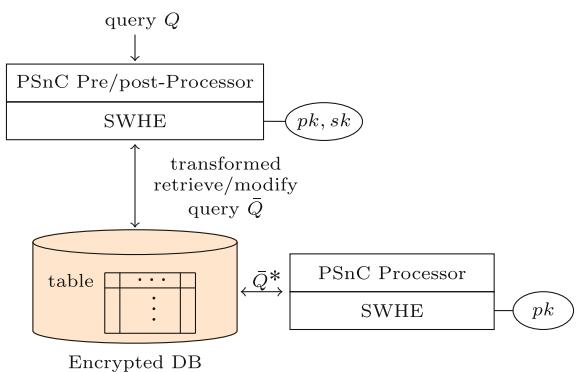


Fig. 1. Our PSnC framework

write queries (*e.g.*, using the `insert-into` statement). Suppose that the user wants to submit a retrieval query Q to the DB server. Before being submitted, the query Q needs to be properly pre-processed so that all clear messages, such as constant values, are encrypted under the public key pk . We denote this transformed query by \bar{Q} .

Upon receiving \bar{Q} , the DB server compiles it into \bar{Q}^* by applying our techniques. The readers can consider a dedicated module for performing this task.¹ Hereafter, we call the module a *Private Search-and-compute* (PSnC) processor. Next, the DB server homomorphically evaluates \bar{Q}^* over the fully encrypted databases and returns the resulting ciphertexts to the user. The DB user can decrypt the output using his private key sk while learning no additional data except for the records satisfying the `where` conditions.

Figure 1 graphically illustrates the high-level architecture of our approach.

1.3 Closely Related Work

A few results closely related to our work can be found in the literature. First, Lauter et al. in [15] showed how to privately compute `avg` and `var` functions using a variant of Brakerski et al.’s SWHE scheme [8]. However, their work only focused on applying homomorphic encryption to compute aggregate functions in query statements. Thus, it is not clear how to address their `where` clauses in a private manner.

Recently, Boneh et al. [4] proposed a way to privately process the `where` clause in a `select` statement and produce a set of matching indices. Their technique uses private set intersection together with homomorphic encryption. It also has the following drawbacks: (1) their scheme only allows conjunctive and disjunctive conditions; (2) the equality test is restricted to comparisons with a constant value; and (3) the users must revisit the server to obtain a list of real tuples because they only know the indices of those tuples.

Our work differs in several ways from prior efforts. First, our solution can privately process the `select` clause and the `where` clause all at once. Second, our solution supports a wide range of query types—from simple search queries to join queries. In particular, our solution allows the DB users to express rich conditions, including $<$, \leq , $>$, \geq , and $<>$.

Organization. The remainder of the paper is structured as follows. In Sect. 2, we briefly review the BGV-type homomorphic encryption scheme. In Sect. 3, we construct the optimized circuits for expressing queries. Then, in Sect. 4, we show how to construct database queries having search and/or compute operations using our circuit primitives. Section 5 presents our optimization techniques for further improvements in performance, and Sect. 6 shows the experimental evaluations of our constructions.

¹ Alternatively, one may imagine that \bar{Q}^* transformed by the DB user directly is sent to the DB server. However, considering optimization and performance, we believe that the better choice involves the module becoming part of the DBMS.

2 Preliminaries

In this section, we focus on describing the efficient variant of the Brakerski-Gentry-Vaikuntanathan(BGV)-type cryptosystem [6, 14], which is our underlying encryption scheme. In what follows, we give a description of the security model that our constructions assume.

2.1 The BGV-Type SWHE Scheme

For a security parameter κ , we choose an $m \in \mathbb{Z}$ that defines the m -th cyclotomic polynomial $\varPhi_m(X)$. For a polynomial ring $\mathbb{A} = \mathbb{Z}[X]/\langle \varPhi_m(X) \rangle$, we set the message space to $\mathbb{A}_t := \mathbb{A}/t\mathbb{A}$ for some fixed $t \geq 2$ and the ciphertext space to $\mathbb{A}_q := \mathbb{A}/q\mathbb{A}$ for an integer q . We choose a chain of moduli $q_0 < q_1 < \dots < q_L = q$ whereby the SWHE scheme can evaluate a depth- L arithmetic circuit. Here is the RLWE-based SWHE scheme:

- $(a, b; \mathbf{s}) \leftarrow \mathsf{Kg}(1^\kappa, h, \sigma, q_L)$: The algorithm Kg chooses a weight h secret key \mathbf{s} and generates a RLWE instance (a, b) relative to that secret key. We set the secret key $sk = \mathbf{s}$ and the public key $pk = (a, b)$.
- $\mathbf{c} \leftarrow \mathsf{E}_{pk}(x)$: To encrypt a message $x \in \mathbb{A}_t$, the algorithm chooses a small polynomial v and two Gaussian polynomials e_0, e_1 (with variance σ^2). It outputs the ciphertext $\mathbf{c} = (c_0, c_1)$ by computing

$$(c_0, c_1) = (x, 0) + (bv + te_0, av + te_1) \bmod q_L.$$

- $x \leftarrow \mathsf{D}_{sk}(\mathbf{c})$: Given a ciphertext $\mathbf{c} = (c_0, c_1)$ at level l , the algorithm outputs $x = [c_0 - \mathbf{s} \cdot c_1]_{q_l} \bmod t$.
- $\mathbf{c}_f \leftarrow \mathsf{Ev}_{ek}(f; \mathbf{c}, \mathbf{c}')$: If the function f is an addition over ciphertexts, the algorithm outputs the ciphertext performed by simple component-wise addition of the two ciphertexts. If f is a multiplication over ciphertexts, it outputs the one performed using a tensor product.

2.2 Security Model

We will consider the following threat model. First, we assume that an SQL server is semi-honest. Thus, it should follow all specifications of our scheme. However, an adversary is allowed to access all databases maintained by a corrupted SQL server. Moreover, a corrupted DBA may become such an attacker. It is fairly plausible for an attacker to legally login to the SQL server, to make an illegal copy of interesting data, and to hand it over to any malicious buyer. Therefore, the DB server should learn nothing about a query beyond what is explicitly revealed (*e.g.*, the number of tuples).

Second, we assume that a DB user is also semi-honest but is not allowed to collude with an SQL server. Some corrupted DB users can create an illegal copy of sensitive data; however, the volume of illegally copied data leaked at any given time is assumed to be negligible. The DB user should not be given access to data that are not part of the query result.

To formulate our security model, we follow Boneh et al.'s security definition [4]. Specifically, the dishonest DB server should not be able to distinguish between \bar{Q}_0 and \bar{Q}_1 , where two transformed queries \bar{Q}_0 and \bar{Q}_1 have the same syntactical form. Moreover, the adversarial DB user should not be able to distinguish two encrypted DBs $\bar{\text{DB}}_0$ and $\bar{\text{DB}}_1$ for every fixed query Q and for all pairs of DBs $(\text{DB}_0, \text{DB}_1)$ such that $Q(\bar{\text{DB}}_0) = Q(\bar{\text{DB}}_1)$.

3 Circuit Primitives

We devise three primitives: equality, comparison (for the `where` clauses) and an integer addition circuit (for the `select` clauses). We focus on a method of optimizing these circuits with respect to the depth and required homomorphic operations. To do this, we make use of SIMD along with automorphism operation.

When input messages are decomposed and encrypted in a bitwise manner, the encryption \bar{x} of a message $x = x_{\mu-1} \cdots x_1 x_0$ means $\{\bar{x}_0, \bar{x}_1, \dots, \bar{x}_{\mu-1}\}$, where $x_i \in \{0, 1\}$. We use “+” to denote homomorphic addition and \mathbf{A} to denote the number of additions. Similarly, for homomorphic multiplication, we use “.” and \mathbf{M} .

3.1 Equality Circuit

For two μ -bit integers x and y , we define an arithmetic circuit for the equality test as follows:

$$\mathbf{equal}(\bar{x}, \bar{y}) = \prod_{i=0}^{\mu-1} (1 + \bar{x}_i + \bar{y}_i). \quad (1)$$

The output of $\mathbf{equal}(\cdot, \cdot)$ is $\bar{1}$ in the case of equality and $\bar{0}$ otherwise. In the bit-sliced implementation, we assume that one ciphertext is used per bit; therefore, we have 2μ ciphertexts in total for evaluating the equality test. Instead of regular multiplication, if we multiply each term after forming a binary-tree structure, the depth of the \mathbf{equal} circuit becomes $\log \mu$. Specifically, the algorithm requires two homomorphic additions for computing $1 + \bar{x}_i + \bar{y}_i$ and that μ ciphertexts be multiplied by each other while consuming $\log \mu$ depth.

Optimizations. Our optimizations are focused on minimizing the number of homomorphic operations, especially for homomorphic multiplication. As shown by Smart and Vercauteren [20], we can pack each bit x_i into a single ciphertext. Next, we expand the right-hand side of Eq. (1) and rearrange each term so as to fit in well with the SIMD executions. Then, we repeatedly apply SIMD operations to a vector of SIMD words. This is the key to reducing the number of homomorphic multiplications from $\mu - 1$ to $\log \mu$. We provide a better description of the complexity in Table 1.

Table 1. Complexity of circuit primitives

	Circuits	Complexity
Depth	<code>equal</code>	$\log \mu$
	<code>comp</code>	$1 + \log \mu$
	<code>fadd</code>	$1 + \log(\nu - 2)$
Comp. ^a	<code>equal</code>	$2A + (\log \mu) M$
	<code>comp</code>	$(\mu + 1 + \log \mu) A + (2\mu - 2) M$
	<code>fadd</code>	$\nu A + (3\nu - 5) M$

^aComp.: Computational complexity during homomorphic evaluations

3.2 Greater-than Comparison Circuit

For two unsigned μ -bit integers, the circuit `comp`(\bar{x}, \bar{y}) outputs $\bar{0}$ if $x \geq y$ and $\bar{1}$ otherwise. This operation can be recursively defined as follows:

$$\text{comp}(\bar{x}, \bar{y}) = \bar{c}_{\mu-1}, \quad (2)$$

where $\bar{c}_i = (1 + \bar{x}_i) \cdot \bar{y}_i + (1 + \bar{x}_i + \bar{y}_i) \cdot \bar{c}_{i-1}$ for $i \geq 1$ with an initial value $\bar{c}_0 = (1 + \bar{x}_0) \cdot \bar{y}_0$.

Optimizations. As the first step of optimization, we express Eq. (2) in the following closed form

$$\bar{c}_{\mu-1} = (1 + \bar{x}_{\mu-1}) \cdot \bar{y}_{\mu-1} + \sum_{i=0}^{\mu-2} (1 + \bar{x}_i) \cdot \bar{y}_i \cdot d_{i+1}d_{i+2} \cdots d_{\mu-1},$$

where $d_j = (1 + \bar{x}_j + \bar{y}_j)$. Because it has degree $\mu + 1$, we can deduce that the depth of the circuit is $\log(\mu + 1)$. Next, it is easy to see that a naive construction of the circuit incurs $\mathcal{O}(\mu^2)$ homomorphic multiplications.

The key observation is that the closed form is expressed by a sum of products of $(1 + \bar{x}_i) \cdot \bar{y}_i$ and $(1 + \bar{x}_i + \bar{y}_i)$ terms for $i \in [0, \mu - 1]$. We are able to compute $(1 + \bar{x}_i) \cdot \bar{y}_i$ for all i using only 1 homomorphic multiplication due to the use of the SIMD technique. Now, we have to compute $\prod_{k=i}^{\mu-1} d_k$ for each $i \in [1, \mu - 2]$. As mentioned above, a naive method incurs $\mathcal{O}(\mu^2)$, but using SIMD operations requires one to perform only $2\mu - 4$ homomorphic multiplications, consuming $\log \mu$ depth. Finally, we need to multiply $(1 + \bar{x}_i) \cdot \bar{y}_i$ by the result of the above computation, which also incurs only 1 homomorphic multiplication. Thus, the total number of homomorphic multiplications equals $2\mu - 2$.

Remark 1. We can address the signed numbers by slightly modifying the circuit. Assume that we place a sign bit in the leftmost position of a value (e.g., 0 for a positive number and 1 for a negative number) and use the two's complement

system. Then, for two μ -bit values x and y , $\text{comp}(\bar{x}, \bar{y}) = \bar{c}_{\mu-1} + \bar{x}_{\mu-1} + \bar{y}_{\mu-1}$. It is clear that the case of two positive numbers corresponds to $\bar{x}_{\mu-1} = \bar{y}_{\mu-1} = \bar{0}$.

3.3 Integer Addition Circuit

Suppose that for two μ -bit integers x and y and for an integer $\nu > \mu$, we construct two ν -bit integers by padding zeros on the left. Then, a size- ν full-adder fadd_ν is recursively defined as follows: $\text{fadd}_\nu(\bar{x}, \bar{y}) = (\bar{s}_0, \bar{s}_1, \dots, \bar{s}_{\nu-1})$ where a sum $\bar{s}_i = \bar{x}_i + \bar{y}_i + \bar{c}_{i-1}$ and a carry-out $\bar{c}_i = (\bar{x}_i \cdot \bar{y}_i) + ((\bar{x}_i + \bar{y}_i) \cdot \bar{c}_{i-1})$ for $i \in [1, \nu-1]$ with initial values $\bar{s}_0 = \bar{x}_0 + \bar{y}_0$ and $\bar{c}_0 = \bar{x}_0 \cdot \bar{y}_0$. The main reason for considering such a large full-adder is to cover SQL aggregate functions with many additions.

Optimizations. Our strategy for optimization is the same as above. Namely, we express each sum and carry in the closed form and find a way to minimize the number of homomorphic operations using SIMD operations. As a result, \bar{s}_i 's are written as follows: $\bar{s}_i = \bar{x}_i + \bar{y}_i + \sum_{j=0}^{i-1} t_{ij}$ where $t_{ij} = (\bar{x}_j \cdot \bar{y}_j) \prod_{j+1 \leq k \leq i-1} (\bar{x}_k + \bar{y}_k)$ for $j < i-1$ and $t_{i,i-1} = \bar{x}_{i-1} \cdot \bar{y}_{i-1}$. When $i = \nu - 1$ and $j = 0$, because $\nu - 2$ homomorphic multiplications are required, we see that the circuit has $\log(\nu - 2)$ depth. However, we need to perform an additional multiplication by $\bar{x}_j \cdot \bar{y}_j$. Thus, the total depth amounts to $\log(\nu - 2) + 1$. As before, the use of SIMD and parallelism by automorphism allows us to evaluate the integer addition circuit with only $3\nu - 5$ homomorphic multiplications, while a naive method requires $(\nu^3 - 3\nu^2 + 8\nu)/6$ homomorphic multiplications.

4 Search-and-Compute on Encrypted Data

In this section, we show how to efficiently perform queries on encrypted data using the circuit primitives. We first describe our techniques in a general setting and then show how our ideas are applied to database applications.

4.1 General-Purpose Search-and-Compute

We begin by describing our basic idea for performing a *search* operation over encrypted data. We assume that a collection of data is partitioned into N μ -bit items denoted by $R_1 \parallel \dots \parallel R_N$ and that the data have been encrypted and stored in the form of $\bar{R}_1 \parallel \dots \parallel \bar{R}_N$.

For a predicate φ on a ciphertext \mathcal{C} , a search on encrypted data outputs \bar{R}_i if $\varphi(\bar{R}_i) = \bar{1}$ and $\bar{0}$ otherwise. More formally, let $\varphi : \mathcal{C} \rightarrow \{\bar{0}, \bar{1}\}$ be a predicate on encrypted data. Then, we say that $S_\varphi : \mathcal{C}^N \rightarrow \mathcal{C}^N$ is a search on the encrypted data and define $S_\varphi(\bar{R}_1, \dots, \bar{R}_N) := (\varphi(\bar{R}_1) \cdot \bar{R}_1, \dots, \varphi(\bar{R}_N) \cdot \bar{R}_N)$.

We then extend this operation to a more general operation on encrypted data, *i.e.*, search-and-compute on encrypted data, as follows. Let $F : \mathcal{C}^N \rightarrow \mathcal{C}$ be an

arithmetic function on encryptions. Then, for restricted search $S_\varphi : \mathcal{C}^N \rightarrow \mathcal{C}^N$, we say that $(F \circ S_\varphi)(\bar{R}_1, \dots, \bar{R}_N)$ is search-and-compute on encryptions.

Further, we measure the efficiency of the search-and-compute operations on encrypted data in Theorem 1. The theorem states that if we can perform a search on encrypted data restricted by φ , which specifies only the equality operator, then the search queries on encrypted data require $N(2A + \log \mu M)$ homomorphic operations in total. If a predicate φ allows one to specify all the comparison operators in the set $\{<, \leq, >, \geq, \neq\}$, then we can perform $S_\varphi(\bar{R}_1, \dots, \bar{R}_N)$ with $\mathcal{O}(\mu N)$ homomorphic multiplications.

Theorem 1. *Let $M(\varphi)$ and $M(F)$ be the total number of homomorphic multiplications for φ and F , respectively. Then, we can perform $(F \circ S_\varphi)(\bar{R}_1, \dots, \bar{R}_N)$ with $\mathcal{O}(N(M(\varphi)) + M(F))$ homomorphic operations. Specifically, we can perform a search on encrypted data restricted by φ using at most $\mathcal{O}(N(M(\varphi)))$ homomorphic operations.*

Proof. Because homomorphic multiplication dominates the performance of the operation, we might only count it. Because a predicate φ requires $\mathcal{O}(M(\varphi))$ homomorphic operations, we see that S_φ requires $\mathcal{O}(N(M(\varphi)))$ homomorphic operations to compute the predicate N times. Then, the operation uses $\mathcal{O}(M(F))$ homomorphic operations to evaluate an arithmetic function F on encrypted data. Therefore, we can conclude that the total computation complexity of search-and-compute on encryptions is $\mathcal{O}(N(M(\varphi)) + M(F))$. In particular, if we consider the search on encrypted data, F can be considered to be the identity map. Therefore, we can perform a search on encrypted data restricted by φ using at most $\mathcal{O}(N(M(\varphi)))$ homomorphic operations. \square

Security. Secrecy against a semi-honest DB server is ensured because encrypted data cannot be leaked due to the semantic security of our underlying SWHE scheme. Secrecy against a semi-honest DB user follows because the result of queries expressed by our circuit primitives is equivalent to $\bar{0}$ if specified conditions do not hold; therefore, the resulting ciphertext is equal to $\bar{0}$. This implies that the evaluated ciphertexts do not leak anything else except for the number of unsatisfied tuples.

4.2 Applications to Encrypted Databases

We denote $R(A_1, \dots, A_d)$ as a relation schema R of degree d consisting of attributes A_1, \dots, A_d , and we denote by \bar{A}_j the corresponding encrypted attribute. As mentioned above, we use $A_j^{(i)}$ to denote the j -th attribute value of the i -th tuple, and for convenience, we assume that each of them has a length of μ bits.

4.2.1 Search Queries

Simple Selection Queries. Consider a simple retrieval query as follows:

```
select Aj1, ..., Ajs
  from R
 where Aj0 = α;
```

(Q.1)

where α is a constant value. An efficient construction of (Q.1) using our `equal` circuit is as follows:

$$\text{equal} \left(\bar{A}_{j_0}^{(i)}, \bar{\alpha} \right) \cdot \left(\bar{A}_{j_1}^{(i)}, \dots, \bar{A}_{j_s}^{(i)} \right)$$
(Q*.1)

for each $i \in [1, N]$. It follows from Theorem 1 that (Q*.1) has the complexity evaluation given in Table 2.

Conjunctive & Disjunctive Queries. The query (Q.1) is extended by adding one or more conjunctive or disjunctive conditions to the `where` clause. Consider a conjunctive query as follows:

```
select Aj1, ..., Ajs
  from R
 where Aj'_1 = α1 and ... and Aj'_τ = ατ;
```

(Q.2)

The query (Q.2) is expressed as the following: For each $i \in [1, N]$,

$$\prod_{k=1}^{\tau} \text{equal} \left(\bar{A}_{j'_k}^{(i)}, \bar{\alpha}_k \right) \cdot \left(\bar{A}_{j_1}^{(i)}, \dots, \bar{A}_{j_s}^{(i)} \right).$$
(Q*.2)

A disjunctive query whose logical connectives are all `ors` is also evaluated by changing the predicate into $\left(1 + \prod_{k=1}^{\tau} \left(\text{equal} \left(\bar{A}_{j'_k}^{(i)}, \bar{\alpha}_k \right) + 1 \right) \right)$. Denoting by τ the number of connectives, Q*.2 additionally requires $\log \tau$ in depth to compute the multiplications among the τ equality tests in comparison with (Q*.1). Table 2 reports the complexity analysis.

Table 2. Complexity of search queries

	Queries	Complexity
Depth	(Q*.1)	$1 + \log \mu$
	(Q*.2)	$1 + \log \mu + \log \tau$
Comp.	(Q*.1)	$2NA + N(1 + \log \mu)M$
	(Q*.2)	$2\tau NA + \tau N(1 + \log \mu)M$

4.2.2 Search-and-Compute Queries

We continue presenting important real constructions as an extension of Theorem 1, in which F is one of the built-in SQL aggregate functions—`sum`, `avg`, `count` and `max`. We begin with the case $F = \text{sum}$.

Search-and-sum Query. Consider the following `sum` query:

```
select sum(Aj1)
  from R
 where Aj0 = α;
```

(Q.3)

As mentioned above, due to our plaintext space being \mathbb{Z}_2 , repeatedly applying simple homomorphic additions does not ensure correctness. This is the motivation for our integer addition circuit (See Sect. 3.3). Now, we can efficiently perform (Q.3), expressed as follows:

$$\text{fadd}_{\mu+\log N} \left(\text{equal} \left(\bar{A}_{j_0}^{(i)}, \bar{\alpha} \right) \cdot \bar{A}_{j_1}^{(i)} \right). \quad (\bar{Q}^*.3)$$

Because the result of the search-and-sum query is less than $2^\mu N$, it suffices to use a full adder of size $\nu = \mu + \log N$ for adding all the values. Using our optimized equality circuit, ($\bar{Q}^*.3$) requires N equality tests in total and N homomorphic multiplications for each result of the test. Thus, the total computation cost is $(2N + \nu(N - 1))\mathbf{A} + (N(1 + \log \mu) + (N - 1)(3\nu - 5))\mathbf{M}$ with the depth $1 + \log \mu + \log N(1 + \log(\nu - 2))$ based on Theorem 2 below.

Theorem 2. *Let $|R|$ denote the cardinality of a set of tuples from a relation schema R . Suppose that all the keyword attributes in the `where` clause and the numeric attributes in the `select` clause have $\|kwd\|$ bits and $\|num\|$ bits, respectively. Then, a search-and-sum query can be processed with the depth*

$$1 + \lceil \log(\|kwd\|) \rceil + \lceil \log |R| \rceil \cdot (1 + \lceil \log(\|num\| + \lceil \log |R| \rceil - 2) \rceil).$$

Proof. The query $\bar{Q}^*.3$ consumes $1 + \lceil \log(\|kwd\|) \rceil$ levels to compute all the equality tests. Then, it performs $(|R| - 1)$ full-adder operations on the results, each of which is of size $(\|num\| + \lceil \log |R| \rceil)$ and which consumes $(1 + \lceil \log(\|num\| + \lceil \log |R| \rceil - 2) \rceil)$ levels. \square

Search-and-Count Query. We observe that search-and-count queries can be processed in a similar manner. For example, assume a search-and-count query with `count(*)` in place of `sum(Aj1)` in (Q.3). The query can also be efficiently processed by $\text{fadd}_{\log N} \left(\text{equal} \left(\bar{A}_{j_0}^{(i)}, \bar{\alpha} \right) \right)$.

Search-and-Avg Query. To process a search-and-compute query with the `avg` aggregate function, it suffices to compute search-and-sum queries because an average can be obtained using one division after decryption.

Search-and-Max(Min) Query. It is clear that one can obtain the `max` (or `min`) aggregate function by repeatedly applying the `comp` circuit primitive.

4.2.3 Join Queries

Now, we design the join queries within the search-and-compute paradigm. Suppose that we have the other relation $S(B_1, \dots, B_e)$ consisting of M tuples for $M \leq N$. First, we consider a simple join query as follows:

```
select r.Aj1, ..., r.Ajs, s.Bj'1, ..., s.Bj's'
  from R as r, S as s
 where r.Ajk = s.Bj'k';
```

(Q.4)

Then this type of query is expressed as the following: For each $i \in [1, N], i' \in [1, M]$,

$$\text{equal} \left(r.\bar{A}_{j_k}^{(i)}, s.\bar{B}_{j'_{k'}}^{(i')} \right) \cdot \left(r.\bar{A}_{j_1}^{(i)}, s.\bar{B}_{j'_1}^{(i')}, \dots \right). \quad (\bar{Q}^*.4)$$

For fixed i and i' , we suppose that each numeric-type attribute is packed in only one ciphertext. Then, the only difference from $(\bar{Q}^*.1)$ is that $(\bar{Q}^*.4)$ requires two homomorphic multiplications by the result of search operations; thus, we need to perform NM equality tests in total. Hence, the depth of circuit needed to process $(\bar{Q}^*.4)$ is $1 + \log \mu$, and the computation complexity is $(2NM)\mathbf{A} + NM(2 + \log \mu)\mathbf{M}$.

Next, we consider an advanced join query (Q.5) with two aggregate functions `sum(r.Aj)`, `count(*)` and the same simple condition as (Q.4). Assuming $\text{sum}(r.A_j) < 2^\mu NM$, we use a full adder of size $\nu = \mu + \log(NM)$. By contrast, the result of $\text{count(*)} < NM$, and it suffices to use a full adder of size $\log(NM)$. Thus, one candidate of circuit construction for (Q.5) is as follows:

$$\begin{aligned} \text{fadd}_{\mu+\log NM} & \left(\text{equal} \left(r.\bar{A}_{j_k}^{(i)}, s.\bar{B}_{j'_{k'}}^{(i')} \right) \cdot r.\bar{A}_j^{(i)} \right) \\ \text{fadd}_{\log NM} & \left(\text{equal} \left(r.\bar{A}_{j_k}^{(i)}, s.\bar{B}_{j'_{k'}}^{(i')} \right) \right). \end{aligned} \quad (\bar{Q}^*.5)$$

With respect to `sum(r.Aj)`, this is the same as $\bar{Q}^*.3$, except for the number of operands for additions. Therefore, the depth for evaluation amounts to $1 + \log \mu + \log(NM)(1 + \log(\nu - 2))$ and the computation complexity is $(2NM + \nu(NM - 1))\mathbf{A} + (NM(1 + \log \mu) + (NM - 1)(3\nu - 5))\mathbf{M}$.

5 Performance Improvements

There is still room to further improve the performance of the circuit primitives in Sect. 3. Our strategies are composed of three interrelated parts: Switch the message space \mathbb{Z}_2 into \mathbb{Z}_t , adapt the circuit primitives to \mathbb{Z}_t , and fine-tune the circuit primitives using SIMD operations again.

5.1 Larger Message Spaces with Lazy Carry Processing

If we encrypt messages in a bit-by-bit manner, the primary advantage is that two comparison operations are very cheap, but running an integer addition circuit on encrypted data is expensive (see Table 3). On the contrary, it would be of substantial benefit to take the message domain as a large integer ring if one can quite efficiently evaluate the addition circuit with much lesser depth. One of the important motivations of using such a large message space is that the bit length of keyword attributes (*e.g.*, ≤ 20 bits) in the `where` clause is generally smaller than that of numeric-type attributes (*e.g.*, ≥ 30 bits) in the `select` clause.

Specifically, if we represent a numeric-type attribute A in the radix 2^ω , then we have

$$\sum_i A^{(i)} = \sum_k \sum_i [A^{(i)}]_k \cdot (2^\omega)^k;$$

therefore, it suffices to compute $\sum_i [A^{(i)}]_k$ over the integers. Assuming that the plaintext modulus t is sufficiently large, we are able to perform addition without overflow in \mathbb{Z}_t . We should note that we only have to process carry operations after computing each of them over the large integer ring.

To verify the performance gained by integer encoding, we report the running time of each circuit primitive in Table 3. We suspect that integer encoding yields more benefits in performing search-and-compute queries because aggregate functions extensively rely on addition.

Table 3. Running-time comparisons in \mathbb{Z}_2 and $\mathbb{Z}_{2^{14}}$

Message space	<code>equal</code>	<code>comp</code>	<code>add</code>
	(10-bits)	(10-bits)	(30-bits)
\mathbb{Z}_2	2.2621 ms	8.5906 ms	228.5180 ms
$\mathbb{Z}_{2^{14}}$	208.6543 ms	307.5200 ms	0.0004 ms

5.2 Calibrating Circuit Primitives

It is clear that the use of a different message space results in modifications of our circuit primitives. Before discussing our modifications in detail, we need to determine some lower bounds of depth for homomorphic multiplication as a function of t . We have two types of homomorphic multiplications: multiplying a ciphertext either by another ciphertext or by a known constant. We formally state this in Theorem 3.

Theorem 3. *Suppose that the native message space of the BGV cryptosystem is a polynomial ring $\mathbb{Z}_t[X]/\langle \Phi_m(X) \rangle$ and that a chain of moduli is defined by a set of primes of roughly the same size, p_0, \dots, p_L , that is, the i -th modulus q_i is defined as $q_i = \prod_{k=0}^i p_k$. For simplicity, assume that p is the size of the p_k 's.*

Let us denote by h the Hamming weight of the secret key. For $i \leq j$, let \mathbf{c} and \mathbf{c}' be normal ciphertexts at level i and j , respectively. Then, the depth, denoted by $\tilde{\mathbf{d}}$, for multiplying \mathbf{c} and \mathbf{c}' is the smallest nonnegative integer that satisfies the following inequality:

$$t^2 \cdot \phi(m) \cdot (1 + h) \cdot ([q_i^{-1}]_t)^2 < 6p^{2 \cdot \tilde{\mathbf{d}}}.$$

In addition, the depth, denoted by $\tilde{\mathbf{d}}_c$, for multiplying \mathbf{c} by a constant is the smallest nonnegative integer for which the following inequality holds:

$$\phi(m) \cdot (t/2)^2 < p^{2 \cdot \tilde{\mathbf{d}}_c}.$$

Proof. Before multiplying two ciphertexts, we set their noise magnitude to be smaller than the pre-set constant $B = t^2\phi(m)(1 + h)/12$ by modulus switching. Subsequently, we obtain a tensor product of the ciphertexts, and the result has the noise magnitude as $2B([q_i^{-1}]_t)^2$. Next, the scale-down is performed by removing small primes p_k 's from the current prime-set of the tensored ciphertext; we say that Δ is the product of the removed primes. We then have $2B^2([q_i^{-1}]_t)^2/\Delta^2 < B$. By assumption, it may be considered that $\Delta = p^{\tilde{\mathbf{d}}}$, which means that $\tilde{\mathbf{d}}$ is the smallest nonnegative integer that satisfies the inequality $2B([q_i^{-1}]_t)^2 < p^{2 \cdot \tilde{\mathbf{d}}}$.

We now consider the case in which \mathbf{c} is multiplied by a constant. As above, the result has approximately the same noise estimate as $B \cdot \phi(m) \cdot (t/2)^2$. Thus, we see that $\tilde{\mathbf{d}}_c$ is the smallest nonnegative integer that satisfies the inequality $\phi(m) \cdot (t/2)^2 < p^{2 \cdot \tilde{\mathbf{d}}_c}$. \square

As a concrete example, we have $\tilde{\mathbf{d}} = 2$ and $\tilde{\mathbf{d}}_c = 1$ in $\mathbb{Z}_{2^{14}}$ with the assumption that $h = 64$ and $m = 13981$.

We now describe a basic idea that underlies our modifications. It is well known that for $x, y \in \{0, 1\}$, the following properties hold:

$$x \oplus y = x + y - 2 \cdot x \cdot y \quad \text{and} \quad x \wedge y = x \cdot y,$$

where $+$, $-$, and \cdot are arithmetic operations over integers. Based on this observation, our equality test can be rewritten as follows:

$$\mathbf{equal}(\bar{x}, \bar{y}) = \prod_{i=0}^{\mu-1} (1 - \bar{x}_i - \bar{y}_i + 2 \cdot \bar{x}_i \cdot \bar{y}_i).$$

We then see that with only a small extra cost, we can construct a new arithmetic circuit for an equality test working on \mathbb{Z}_t . Next, consider the **comp** circuit on \mathbb{Z}_t . Recall that the closed form of $\bar{c}_{\mu-1}$ is

$$\bar{c}_{\mu-1} = (1 - \bar{x}_{\mu-1}) \cdot \bar{y}_{\mu-1} + \sum_{i=0}^{\mu-2} (1 - \bar{x}_i) \cdot \bar{y}_i \cdot (d_{i+1} d_{i+2} \cdots d_{\mu-1}).$$

Rather than $d_j = (1 + \bar{x}_j + \bar{y}_j)$, we set $d_j = (1 + 2 \cdot \bar{x}_j \cdot \bar{y}_j - \bar{y}_j - \bar{x}_j) \cdot (1 + 2 \cdot \bar{x}_j \cdot \bar{y}_j - 2\bar{y}_j)$. As a result, Table 4 shows the complexity results of the search-and-compute queries on encrypted databases of N tuples with μ -bit attributes from using the new message space \mathbb{Z}_t .

Table 4. Complexity of search-and-sum queries

	Search	Complexity
Depth	<code>equal</code>	$(2 + \log \mu) \tilde{d} + \tilde{d}_c$
	<code>conj_τ</code>	$(2 + \log \mu + \log \tau) \tilde{d} + \tilde{d}_c$
	<code>comp</code>	$(4 + \log \mu) \tilde{d} + \tilde{d}_c$
Comp.	<code>equal</code>	$(4N - 1) A + N(3 + \log \mu) M$
	<code>conj_τ</code>	$((3\tau + 1)N - 1) A + \tau N(3 + \log \mu) M$
	<code>comp</code>	$(N(\mu + 5 + \log \mu) - 1) A + N(2\mu + 1) M$

6 Experimental Results

This section demonstrates the performance of query processing expressed by our optimized circuit primitives. The essential goal of the experiments in this section is to verify the efficiency of our solution in terms of performance.

All experiments reported in our paper were performed on a machine with an Intel Xeon 2.3 GHz processor with 192 GB of main memory running a Linux 3.2.0 operating system. All methods were implemented using the GCC compiler version 4.2.1. In our experiments, we used a variant of a BGV-type SWHE scheme [14] with Shoup’s NTL library [18] and Shoup-Halevi’s HE library [19]. Throughout this section, when we measured the average running times, we excluded computing times used in data encryption and decryption.

6.1 Adjusting the Parameters

Without a loss of generality, we assume that the bit length of keyword attributes in the `where` clause is 10-bit and that of numeric-type attributes in the `select` clause is 30-bit. The keyword attributes are expressed in a bit-by-bit manner, and each bit is an element of \mathbb{Z}_{2^r} . In addition, numeric-type attributes are expressed by the radix 2^ω but are still in the same space \mathbb{Z}_{2^r} .

We begin by observing the following relation among the parameters. At this point, we consider the *selectivity* of a selection condition, which means the fraction of tuples that satisfies the condition, and we denote it by ε .

Theorem 4. *Let A be a numeric-type attribute. For a positive integer $\omega \geq 1$, suppose that each attribute is written as $A = \sum_k [A]_k \cdot (2^\omega)^k$ with $0 \leq [A]_k < 2^\omega$. Then, to process a search-and-sum query, one can take a plaintext modulus with $r = iTheta(\omega + \log(\varepsilon N))$.*

Proof. The goal of the theorem is to provide a bound for the size of a plaintext modulus; therefore, we simply omit an overhead bar for all variables. Let us denote by φ a predicate on encrypted data and by A^* a keyword attribute. Then, a search-and-sum query can be written as

$$\sum_i S_\varphi(A^*, \alpha) \cdot A^{(i)} = \sum_k \left(\sum_i S_\varphi(A^*, \alpha) \cdot [A^{(i)}]_k \right) \cdot (2^\omega)^k.$$

We then have that $\sum_i S_\varphi(A^*, \alpha) \cdot [A^{(i)}]_k < 2^\omega \sum_i S_\varphi(A^*, \alpha) = 2^\omega \cdot (\varepsilon N)$. Thus, for a database with N records, it is sufficient to choose r such that $2^\omega \cdot (\varepsilon N) \leq 2^r$. Note, the larger we make the plaintext modulus 2^r , the more noise there is in the ciphertexts and thus the faster we consume the ciphertext level. Therefore, it appears that $\omega + \log(\varepsilon N)$ is the tight bound for the parameter r . \square

One may wonder why $S_\varphi(\cdot, \dots)$ does not take multiple keyword attributes in the proof. Because we consider the selectivity ratio, it does not need to do so. In our experiments, we varied the selectivity ratio from 5 to 40 % and plotted the average running time of queries over a database with $N = 10^2, 10^3$, and 10^4 tuples.

6.2 Experiments for Search

We measured the running time per query while varying the number of numeric-type attributes. We take the ring modulus $m = 8191$, and each of the ciphertexts has 630 plaintext slots. For $N = 1$, the experiment of $(\bar{Q}^*.1)$ query is given in

Table 5. Experiment results

(a) $(\bar{Q}^*.1)$ and $(\bar{Q}^*.2)$

Message Space	τ	L	s	Timing
\mathbb{Z}_2	1	6	5	0.38s
			10	0.76s
			20	1.51s
\mathbb{Z}_2	4	7	5	2.04s
			10	4.09s
			20	8.17s

(b) Case I

N	ε	Message Space	Radix	L	Timing
10^2	< 16%	$\mathbb{Z}_{2^{14}}$	2^{10}	14	3.69s
	< 32%			15	3.89s
10^3	$\leq 6\%$	$\mathbb{Z}_{2^{16}}$	2^{10}	15	38.78s
	$\leq 25\%$			2^8	51.64s
10^4	$\leq 10\%$	$\mathbb{Z}_{2^{16}}$	2^6		681.05s
	$\leq 20\%$			2^5	817.26s
	$\leq 40\%$			2^4	1089.68s

the top three rows of Table 5a and that of $(\bar{Q}^*.2)$ is in the bottom three rows in Table 5a, where s is the number of attributes and L is the number of ciphertext moduli.

6.3 Experiments for Search-and-Sum

We conducted a series of additional experiments to measure performance of search-and-sum queries. Because each of the ciphertexts can hold ℓ plaintext slots of elements in \mathbb{Z}_{2^r} and because a numeric-type attribute with a length of 30 bits is encoded into $\tilde{\omega}$ ($= \lceil 30/\log(2^\omega) \rceil = \lceil 30/\omega \rceil$) slots, we can process $\tilde{\ell}$ ($= \lfloor \ell/\tilde{\omega} \rfloor$) attributes per ciphertext. At first glance, a larger ω seems to be better. However, if ω is too large, by Theorem 4, a plaintext modulus 2^r becomes large. This results in an increased depth of circuits. Therefore, we need to choose a sufficiently large ω whereby the resulting plaintext space is not too large.

We divided our experiment into four cases by types of predicates: (1) Single equality, (2) Single comparison, (3) Multiple equality, and (4) Multiple comparison. In this paper, we only report the experiment results of Case I in Table 5b. We recommend that the readers review the original reference [9] for other experiments in more details.

Case I: Single Equality. This case contains one equality test in the `where` clause. We chose a plaintext space so that the number of plaintext slots is divisible by 10. Then, the entire keyword attribute is packed in only one ciphertext. We used $m = 13981$ so that each of the ciphertexts holds 600 plaintext slots.

Acknowledgements. The authors would like to thank the anonymous reviewers of WAHC 2015 for their valuable comments. Jung Hee Cheon and Miran Kim were supported by Samsung Electronics, Co., Ltd. (No. 0421-20140013). Myungsun Kim was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2014R1A1A2058377).

References

1. Ada Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing. In: Wobber, T., Druschel, P. (eds.) SOSP, pp. 85–100 (2011)
2. Bellare, M., Boldyreva, A., O’Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
4. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D.J.: Private database queries using somewhat homomorphic encryption. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 102–118. Springer, Heidelberg (2013)

5. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325 (2012)
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) FOCS, pp. 97–106 (2011)
8. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
9. Cheon, J.H., Kim, M., Kim, M.: Search-and-compute on encrypted data. IACR Cryptology ePrint Archive, 2014(812) (2014)
10. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. *J. Comput. Secur.* **19**(5), 895–934 (2011)
11. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
12. Feigenbaum, J., Merritt, M.: Open questions, talk abstracts, and summary of discussions. *DIMACS Ser. Discrete Math. Theor. Comput. Sci.* **2**, 1–45 (1991)
13. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178 (2009)
14. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
15. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Cachin, C., Ristenpart, T. (ed.) CCSW, pp. 113–124 (2011)
16. 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)
17. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 165–179 (1978)
18. Shoup, V.: NTL: A library for doing number theory (2009). <http://www.shoup.net/ntl/>
19. Shoup, V., Halevi, S.: Design and implementation of a homomorphic-encryption library. Technical report, IBM Technical Report (2013)
20. Smart, N., Vercauteren, F.: Fully homomorphic SIMD operations. IACR Cryptology ePrint Archive, 2011(133) (2011)
21. Song, D., Wagner, D., Perrig, A.: Practical techniques for searching on encrypted data. In: IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
22. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

Accelerating SWHE Based PIRs Using GPUs

Wei Dai^(✉), Yarkin Doröz, and Berk Sunar

Worcester Polytechnic Institute, Worcester, USA

wdai@wpi.edu

Abstract. In this work we focus on tailoring and optimizing the computational Private Information Retrieval (cPIR) scheme proposed in WAHC 2014 for efficient execution on graphics processing units (GPUs). Exploiting the mass parallelism in GPUs is a commonly used approach in speeding up cPIRs. Our goal is to eliminate the efficiency bottleneck of the Doröz et al. construction which would allow us to take advantage of its excellent bandwidth performance. To this end, we develop custom code to support polynomial ring operations and extend them to realize the evaluation functions in an optimized manner on high end GPUs. Specifically, we develop optimized CUDA code to support large degree/large coefficient polynomial arithmetic operations such as modular multiplication/reduction, and modulus switching. Moreover, we choose same prime numbers for both the CRT domain representation of the polynomials and for the modulus switching implementation of the somewhat homomorphic encryption scheme. This allows us to combine two arithmetic domains, which reduces the number of domain conversions and permits us to perform faster arithmetic. Our implementation achieves 14–34 times speedup for index comparison and 4–18 times speedup for data aggregation compared to a pure CPU software implementation.

Keywords: Private information retrieval · Homomorphic encryption · NTRU

1 Introduction

A Private Information Retrieval (PIR) permits Alice to store a database D at a remote server Bob with the promise that Alice can retrieve $D(i)$ without revealing i or $D(i)$ to Bob. An information theoretic PIR scheme was first introduced in [1] where Bob’s knowledge of i was limited using information theoretic arguments. Chor and Gilboa [2,3] introduced the concept of computational PIRs (cPIR). In cPIR, Alice is content to have Bob facing instead a computationally *difficult* problem to extract any significant information about i or $D(i)$. In [4] Kushilevitz and Ostrovsky presented the first single server PIR scheme based on the computational difficulty of deciding the quadratic residuosity of a number modulo a product of two large primes. Other cPIR constructions include [5] which is based on the computational difficulty of deciding whether a prime p divides $\phi(m)$ for any composite integer m of unknown factorization where $\phi()$

denotes Euler's totient function. In [6] another cPIR scheme was presented that generalizes the scheme in [5] while using a variation on the security assumption. The construction in [6] achieves a communication complexity of $O(k + d)$ where k is the security parameter satisfying $k > \log(N)$, N is the database size, and d is the bit-length of the retrieved data. In [7] Lipmaa presented a different cPIR scheme that employs an additively homomorphic encryption scheme with improved communication performance. Later in [8], an efficient PIR scheme was constructed using a partially homomorphic encryption algorithm. The first lattice based cPIR construction was proposed by Aguilar-Melchor and Gaborit [9]. Olumofin and Goldberg [10] revisited the performance analysis and found that the lattice-based PIR scheme by Aguilar-Melchor and Gaborit [9] to be an order of magnitude more efficient than the trivial PIR. These schemes utilize a combination of clever approaches and a diverse set of tools to construct cPIR schemes. Clearly given a fully or somewhat homomorphic encryption (FHE or SWHE) scheme achieving a cPIR construction would be conceptually trivial. With the recent advances and renewed interest in homomorphic encryption, new FHE schemes [11–15] and optimizations such as modulus and key switching [16], batching and SIMD optimizations [17] have become available. The more recent work by Doröz, Sunar and Hammouri [18] leveraged an NTRU based leveled SWHE scheme along with optimizations to construct an efficient cPIR. The rather simple cPIR construction has excellent bandwidth performance compared to previous implementations, i.e. about three orders of magnitude. However, to enable SWHE evaluation the scheme uses large parameters and therefore the computational cost is excessively higher than traditional PIR constructions, i.e. about 1-2 orders of magnitude.

Our Contribution. We present an Nvidia GPU implementation of a cPIR scheme based on SWHE proposed by Doröz et al. We develop optimized CUDA code to support large degree/large coefficient polynomial arithmetic operations such as modular multiplication/reduction, and modulus switching. For efficiency, we utilize number theoretical transform (NTT) based polynomial multiplication while the operands are kept in the CRT domain representation. The CUDA arithmetic library is then used to implement the two modes of the Doröz et al. cPIR [18]. While the bandwidth requirements are the same as in [18] our implementation is significantly faster. For instance, for a database size of 64 K entries, our index comparison implementation is about 33 times faster while the data aggregation operation is 18 times faster than the implementation of [18].

2 Background

In this section, we briefly explain the Doröz-Sunar-Hammouri (DSH) SWHE based PIR construction. First, we give details about the NTRU based SWHE scheme proposed by López-Alt, Tromer and Vaikuntanathan (ATV) [15]. Second, we explain the PIR construction proposed by Doröz et al. which is based on ATV-SWHE scheme.

ATV-SWHE Scheme: The NTRU scheme [19] was originally proposed as a public key encryption scheme. It was later modified by Stehlé and Steinfeld [20] to create a variant whose security is based on the ring learning with error (RLWE) problem. With a number of optimizations, López-Alt, Tromer and Vaikuntanathan extended the construction to a multi-key FHE scheme [15].

Döröz, Sunar, and Hammouri [18] tailored the ATV-SWHE to construct an efficient cPIR. Here we briefly summarize the construction: The polynomials are sampled from a probability distribution χ on B -bounded polynomials in $R_q := \mathbb{Z}_q[x]/(x^n + 1)$ where a polynomial is “ B -bounded” if all of its coefficients lie in $[-B, B]$. The sampled polynomials are used to compute public/secret keys. The scheme can support XOR and AND operations using polynomial additions and multiplications respectively. Noise grows significantly in AND operations, so after each AND operation noise is managed using a technique called modulus switch that is introduced by Brakerski, Gentry and Vaikuntanathan [12]. In modulus switch, a decreasing sequence of moduli $q_0 > q_1 > \dots > q_d$ are selected for each level of the circuit. Also each modulus holds following property: $q_i | q_{i+1}$. The primitive functions of the DSH scheme is as follows:

- **KeyGen:** Choose m -th cyclotomic polynomial $\Phi_m(x)$ of degree $n = \varphi(m)$ as the modulus polynomial. Sample u and g from the distribution χ and compute $f = 2u + 1$ and $h = 2g(f)^{-1}$ in ring $R_{q_i} = \mathbb{Z}_{q_i}[x]/\langle \Phi(x) \rangle$.
- **Encrypt:** Sample s and e from χ distribution and compute $c = hs + 2e + b$, where $b \in \{0, 1\}$ is message and h is public key.
- **Decrypt:** Message m is computed by: $m = cf^{(i)} \pmod{2}$. The $f^{(i)}$ corresponds to private key for i^{th} level that holds: $f^{(i)} = f^{2^i} \in R_{q_i}$.
- **XOR:** The addition of two ciphertexts $c_1 = \text{Enc}(b_1)$ and $c_2 = \text{Enc}(b_2)$ corresponds to XOR operation, i.e. $c_1 + c_2 = \text{Enc}(b_1 \oplus b_2)$.
- **AND:** The multiplication of two ciphertexts corresponds to AND operation, i.e. $c_1 \times c_2 = \text{Enc}(b_1 \cdot b_2)$. After each multiplication, noise level is controlled by applying modulus switch: $\tilde{c}(x) = \left\lfloor \frac{q_{i+1}}{q_i} \tilde{c}(x) \right\rfloor_2$.

DSH-PIR Scheme: For a given database D with $|D| = 2^\ell$ rows and given index $x \in \{0, 1\}^\ell$ we may retrieve data d_x as follows: $\sum_{y \in [2^\ell]} (x = y) d_y \pmod{2}$, where y is an index of the database. The equality check $(x = y)$ is computed using the bits of the indices as: $\prod_{i \in [\ell]} (x_i + y_i + 1)$. Basically, we compare if the bits of x are same with the bits of y for the same positions. If all the bits are same, the product yields a 1. Otherwise, the product evaluates to a 0. Therefore, we can retrieve d_y by computing the sums of products between the comparison results and the corresponding data d_y entries. In the protocol, the bits of the search index x_i are given in encrypted form while the comparison index y_i is in cleartext. Therefore, the $(x = y)$ circuit has to compute the product of ℓ polynomials, i.e. $\prod_{i \in [\ell]} (\text{Enc}(x_i) + y_i + 1)$. This is evaluated with a depth $\log_2(\ell)$ circuit by using a binary tree. Furthermore, the scheme may be extended into a symmetric PIR in which also data is encrypted. In other words, we may encrypt the data as $\text{Enc}(d_y)$ and multiply it with the corresponding ciphertext. This increments the depth of circuit level $\log_2(\ell) + 1$.

Query Modes: Doröz et al. [18] propose two query modes: Single Query and Bundled Query by two complementary uses of a batching technique introduced by Smart and Vercauteren [16, 17]. The technique relies on the CRT where a pack of message bits are encoded into a single binary polynomial using inverse-CRT. This allows us to evaluate a circuit on multiple independent data inputs simultaneously by embedding them into the same ciphertext. The working mechanisms of the two query protocols is as follows:

- **Bundled Query.** In Bundled Query, a client creates multiple queries and batches them into a ciphertext, so the server can process multiple requests at a time. First, the client prepares multiple queries $\beta[j]$, which $j \in \varepsilon$ and ε is the total message slot number, with the following bit representation $\{\beta_{\ell-1}[j], \dots, \beta_0[j]\}$. Then, the client encrypts and encodes the message polynomials for each bit index as: $\tilde{\beta}_i(x) = \text{Enc}(\text{CRT}^{-1}(\beta_i[1], \beta_i[2], \dots, \beta_i[\varepsilon]))$. For each bit location we have a ciphertext which is ℓ in our case. Once the ciphertexts $\tilde{\beta}_i(x)$ are ready, they are sent to the server. The server computes the PIR using the formula: $r(x) = \sum_{y \in [2^\ell]} \left(\prod_{i \in [\ell]} (\tilde{\beta}_i(x) + y_i(x) + 1) \right) d_y(x)$. Here $y_i(x)$ is the batched and encoded row index bits $\{y_i, y_i, \dots, y_i\}$, which results in a single bit result $y_i(x) = y_i$, since each message bit has same value. Once the $r(x)$ is evaluated the server sends the result to the client who then decrypts and decodes the ciphertext and forms a list of retrieved values $\{d_0, d_1, \dots, d_{\varepsilon-1}\} = \text{Decode}(\text{Dec}(r(x)))$.
- **Single Query.** In this mode, the client only gives one index query. The server encodes the indices and the database entries to perform comparison operations for different entries in parallel for same index query. Basically, the client takes the bits of a query β , encrypts each bit $\tilde{\beta}_i(x) = \text{Enc}(\beta_i)$ and sends the ciphertexts to the server. The server computes the PIR operation: $r(x) = \sum_{y \in [2^\ell]} \left(\prod_{i \in [\ell]} (\tilde{\beta}_i(x) + y_i(x) + 1) \right) d_y(x)$. However, this time $y_i(x)$ and $d_y(x)$ are binary polynomials since we are comparing a single query to a block of entries. The term $y_i(x)$ is computed by encoding bits at the same locations of different entries, i.e. $y_i(x) = \text{inverse-CRT}\{y_i[1], \dots, y_i[\varepsilon]\}$. Similarly, d_y is also equal to $\text{inverse-CRT}\{d_0, \dots, d_{\varepsilon-1}\}$. The process compares ε indices simultaneously in each iteration which decreases the overall runtime of the scheme. Once $r(x)$ is computed, the client receives the computed ciphertext which he then decrypts and decodes. If all the bits on the message slots are zero, than $d_y = 0$ and $d_y = 1$ otherwise.

3 GPU Implementation

Here we present an overview of the NTRU based PIR protocol implementation on CUDA GPUs. The client sends an encrypted query to the server. The server homomorphically evaluates the retrieval request, and returns a single ciphertext to the client. We optimize the scheme to better fit our target GPU device, i.e. NVIDIA GeForce GTX690. Nevertheless, the parallelization and optimization

techniques we employ here should work on other GPUs as well. Our GPU device consists of two GK104 chips where each chip holds 1536 cores and 2 GBytes of memory. We make use of both chips and almost evenly distribute workload to the two chips. The performance can be easily improved with a multi-GPUs setup, since the algorithm we present here has a high degree of parallelism.

Platform Overview and Design Strategy. A GPU based server or cluster consists of multiple GPUs. Every GPU is an efficient many-core processor system designed to reach high performance by exploiting parallelism in the computational task. Each *core* can execute a sequential thread. All cores in the same group execute the same instruction at the same time. With a GPU with *warp size* of 32, the code is executed in groups of 32 threads. Threads are further grouped into blocks. All threads in the same block are executed on a single multiprocessor, and therefore are able to share a single software data cache and memory. With general-purpose computing on GPUs, a portion of code that can achieve significant speedup when executed in parallel runs on the GPU, while other code remains on the CPU. Basically, data sets are sent to GPU memory, processed on the GPU and returned from GPU memory. This will pay off as long as the speedup of processing on GPU over on CPU outweighs the latency introduced by the input and output transfers. When GPU cores are handling tasks, besides the computation, memory access latencies also limit the performance gain. Memory on GPU is classified into several types which we list according to the access latency from low to high along with sizes for the Nvidia GeForce GTX 690 GPU: constant memory (64 KB), shared memory (48 KB per block), and global memory (4 GB). Given the drastically different make up of our target platform, it becomes clear that when translating our algorithms into GPU code we need to create a high degree of parallelism and minimize dependencies between data entries to take advantage of the multi-core architecture. Also data transfers between GPU and CPU needs to be reduced to a minimum. In our target application, we are processing a PIR database. The PIR database information is preloaded into the GPU memory. With a database index of b bits (e.g. $b = 32, 16$ or 8 bits), we send the query packaged into b ciphertexts to the server. After retrieval the query returns a single ciphertext per database entry bit back to the client. The entire retrieval computation is performed on the GPUs. For polynomial coefficient-wise operations, since we chose a large polynomial size ($n = 4096, 8190$ or 16384), we achieve a significant level of parallelism without any effort simply due to the way parameters are selected for security. For polynomial operations we introduce two conversions: CRT and NTT. CRT is able to divide any type of polynomial operation into independent operations. NTT is crucial to efficiently support parallel large polynomial multiplication.

GPU Memory. We store polynomials as 1D arrays of integers. To prevent memory contention between the read and write operations by the kernels, we pre-allocate memory pools on GPUs and divide them into smaller chunks with enough space for the CRT domain polynomials. We not only avoid the overhead associated with frequently allocating memory, but also can reclaim memory and reduce the overall memory usage. We always use the faster memory type

available, minimize the global memory access in the kernels, and adjust the number of threads per block to match the shared memory size. We store some data used to generate database indices in advance, since it is constant and takes only 64 MBytes in our largest setting. The input/output data transfer latencies are partially hidden behind database index generation and inverse-CRT operations. We use shared memory as data buffers in CRT and NTT kernels to ensure coalesced memory access.

Mapping the PIR Computation to CUDA Kernels. As described earlier, the Doröz et al. PIR [18] computation is $\sum_{y \in [2^\ell]} \left[\prod_{i \in [\ell]} (x_i + y_i + 1) \right] d_y$. The query retrieval scheme mainly consists of polynomial multiplications over R_q . In addition, as circuit level increases, a modular reduction operation on ciphertexts, i.e. modulus switching, is performed on the polynomial coefficients. For ℓ bits of data, there are $(\ell - 1)$ multiplications and $(\ell - 1)$ modular reductions. We aim at optimizing these two operations. The input ciphertexts are polynomials in the ring R_q . The size of q is very large according to our parameter selection (e.g. 512 bits). We use CRT to convert large polynomials with large integer coefficients into a set of large polynomials with coefficient size small enough to permit streamlined processing. In the transformation we use a series of prime moduli p_1, p_2, \dots, p_l . The result is recovered by computing the CRT inverse.

Chinese Remainder Theorem. With l prime numbers p_1, p_2, \dots, p_l , we obtain a set of independent polynomials from the polynomial $a(x) \in R_q$ as $a(x) = \text{CRT}^{-1}\{a[1](x), a[2](x), \dots, a[l](x)\}$. If $a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0$, we have: $a_i = \text{CRT}^{-1}\{a_i[1], a_i[2], \dots, a_i[l]\}$, $i \in \mathbb{Z}_n$. Since the inverse-CRT returns a result in $\mathbb{Z}_{p_1 p_2 \dots p_l}$ instead of \mathbb{Z}_q , modular reduction on coefficients is needed. Wang et al. [21] introduced a large integer modular reduction implementation on GPUs. Given that a single modular reduction costs multiple threads, processing all the coefficients in a polynomial would be inefficient. In [13], the authors proposed a way to combine inverse-CRT and modulo q reduction steps. We generate q as the product of a sequence of prime numbers and use the prime numbers for CRT: $q_i = \prod_{j=1}^{l-i} p_j$, $i < j$. The size of the primes, should be large enough to control the noise growth during homomorphic evaluations. The upper bound on the prime numbers depends on the polynomial multiplication scheme which will be explained later. After the initial CRT conversion on the input ciphertexts, all the computations are performed in the CRT domain, until we have to compute the inverse-CRT on the output ciphertexts. Focusing more closely on inverse CRT operation, we notice that after the inversion is carried out the coefficients remain in $\mathbb{Z}_{p_1 p_2 \dots p_{l-i}} = \mathbb{Z}_{q_i}$, which means that modulo q reduction is no longer needed. We can also obtain modulo q_j result from a coefficient in \mathbb{Z}_{q_i} for $j > i$. For all $z = \text{CRT}^{-1}\{z[1], z[2], \dots, z[l-i]\} \in \mathbb{Z}_{q_i}$, given that $j > i$, we have $z \pmod{q_j} = \text{CRT}^{-1}\{z[1], z[2], \dots, z[l-j]\}$.

Polynomial Multiplication. In [22], the authors present an efficient polynomial multiplication algorithm on CUDA GPUs. They basically follow Strassen's scheme [23] to multiply two polynomials $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$. Consider the n coefficients of a polynomial as elements in a 0 padded

array of $2n$ elements: $a = \{a_0, a_1, \dots, a_{n-1}, 0, \dots, 0\}$ and $b = \{b_0, b_1, \dots, b_{n-1}, 0, \dots, 0\}$. Perform $2n$ -point NTT on a and b to obtain arrays $A = \text{NTT}(a)$ and $B = \text{NTT}(b)$. Compute the element-wise product $C = A \cdot B$. Finally recover $c = \text{NTT}^{-1}(C)$, in which the elements are the coefficients of $c(x) = a(x) \cdot b(x)$. Four-step Cooley-Tukey NTT iterations [24] are adopted for a fast NTT computation and hence create parallelism for CUDA GPU processors. A special prime number $P = 0xFFFFFFFF00000001$ is chosen for better performance [25]. Since P should be larger than the possible largest coefficient of the polynomial product, we have our limit for the size of CRT prime numbers: $p_i < \sqrt{P/n}$, ($i = 1, 2, \dots, l$).

Polynomial Reduction. In the generic NTRU scheme all polynomial operations are performed in $R_q = \mathbb{Z}_q/\langle \Phi_m(x) \rangle$. Polynomial multiplication is therefore followed by a polynomial reduction. The cyclotomic polynomial modulus $\Phi_m(x)$ is a factor of the special polynomial $(x^m - 1)$. We need to perform a polynomial reduction after every polynomial multiplication with Barrett Reduction which by itself costs three polynomial multiplications. Instead we keep the product in $R'_q = \mathbb{Z}_q/\langle x^m - 1 \rangle$ form during query retrieval and only further (fully) reduce the polynomials to $R_q = \mathbb{Z}_q/\langle \Phi_m(x) \rangle$ in decryption. Polynomial reduction can be achieved by $c(x) \pmod{x^m - 1} = c(x) \pmod{x^m} + c(x)/x^m$. However, the latter method could possibly increase the size of polynomial operands greatly. For instance, with parameters $(n, m) = (16384, 21845)$, we have to conduct 65536-point NTT to multiply two 21845-degree polynomials, instead of 32768-point NTT for 16384-degree multiplication. Nevertheless, assuming the overhead of single sized multiplication is T , double sized NTT takes about $2T$, which is better than the method with Barrett Reduction [26] that takes more than $4T$ in total. Moreover, with parameters $(n, m) = (8190, 8191)$, we can still use the 8192-point NTT.

Modulus Switching. In the NTRU based SWHE, when the circuit level increases, e.g. from level i to level $(i+1)$, polynomials should be scaled from ring R_{q_i} to ring $R_{q_{i+1}}$ without disturbing the message embedded within the ciphertext. This requires a modular reduction operation on coefficients, namely modulus switching. For a ciphertext $c \in R_{q_i}$ at level i , obtain $c' \in R_{q_{i+1}}$ at level $(i+1)$ as follows. First compute $c' = c \pmod{2}$. Note that c' will be close to $c \cdot \frac{q_{i+1}}{q_i}$. This operation is coefficient independent, hence can be executed in parallel. We explain the procedure on a single coefficient. Let a be the target coefficient in \mathbb{Z}_q . One way is to first compute $a' = \lfloor a \cdot \frac{q_{i+1}}{q_i} \rfloor$. Then if $a' \neq a \pmod{2}$, add or subtract 1 for a' to satisfy the equality. The method requires a to stay in \mathbb{Z}_{q_i} . Since in our implementation all operands are kept in the CRT domain, with a straightforward implementation we would have to call the expensive inverse-CRT in every level of the circuit. A technique to avoid the conversion by performing modulus switching in the CRT domain was proposed in [13]. Since $q_i = \prod_{j=1}^{l-i} p_j$ is the product of a sequence of CRT prime numbers, the first step of the previous method can be represented as $a' = \lfloor \frac{a}{p_{l-i}} \rfloor + \epsilon$, $\epsilon \leftarrow \{-1, 0, 1\}$. If $r = a \pmod{p_{l-i}}$, we have $a' \cdot p_{l-i} = a - r + \epsilon \cdot p_{l-i}$. Let $a^* = r - \epsilon \cdot p_{l-i}$. If and only if a^* is even, $a' = a \pmod{2}$. Therefore, $a' = \frac{a-a^*}{p_{l-i}} \in \mathbb{Z}_{q_{i+1}}$ is the result and we only need a^* . Starting from $a = \text{CRT}^{-1}\{a[1], a[2], \dots, a[l-i]\}$, let $a^* = a \pmod{p_{l-i}} = a[l-i]$.

If a^* is odd, add or subtract p_{l-i} to a^* so that $a^* \in (-p_{l-i}, 2p_{l-i})$ is even. Let $a'[j] = (a[j] - a^*)/p_{l-i} \pmod{p_j}$, $j = 1, 2, \dots, l - i - 1$. Then we have $a' = \text{CRT}^{-1}\{a'[1], a'[2], \dots, a'[l - i - 1]\}$ as the new coefficient in $\mathbb{Z}_{q_{i+1}}$.

Algorithm 1. Modulus Switching on Coefficients

Input: Coefficient $a = \text{CRT}^{-1}\{a[1], a[2], \dots, a[l - i]\}$ from level i
Output: Coefficient $a' = \text{CRT}^{-1}\{a'[1], a'[2], \dots, a'[l - i - 1]\}$ for level $(i + 1)$

```

1:  $a^* \leftarrow a[l - i]$ 
2: if  $a^* = 1 \pmod{2}$  then
3:   if  $a^* > (p_{l-i} - 1)/2$  then
4:      $a^* \leftarrow a^* - p_{l-i}$ 
5:   else
6:      $a^* \leftarrow a^* + p_{l-i}$ 
7:   end if
8: end if
9: for  $j \leftarrow 1$  to  $l - i - 1$  do
10:    $a'[j] \leftarrow (a[j] - a^*)/p_{l-i} \pmod{p_j}$ 
11: end for

```

CUDA Kernels on Devices. As described earlier, the GPU devices receive b ciphertexts. Let d be the number of circuit levels for computing the product of the ciphertexts. Using a binary tree: $2^{d-1} < b \leq 2^d$. In Fig. 1, we show the process of computing a single product term of $\sum_{y \in [2^\ell]} \left[\prod_{i \in [\ell]} (x_i + y_i + 1) \right] d_y$ on a GPU for $b = 32$. After the last step an additional multiplication operation is required for generating the response. This consists of either a single multiplication or multiple multiplications depending on bundled query or single query mechanism. Ignoring the last step, we have $(b - 1)$ polynomial multiplications or one binary tree of d depth, and $(b - 1)$ modulus switchings:

- **Multiplication.** First we convert the polynomials to the CRT domain with l prime numbers. In the first level we have $(l \times b)$ polynomials. In the subsequent levels of the computation tree, l is decremented after each modulus switching operation. The number of parallel computation threads is initialized with b and is reduced by half after each multiplication level. At the end we obtain a $((l - d) \times 1)$ polynomials. Ideally we would like to distribute the workload evenly to all devices. Since the Nvidia GeForce GTX 690 only has two GPUs, each device is provided with $(l \times (b/2))$ polynomials to process. Until the last final multiplication the devices work independently.
- **Modulus Switching and Reduction.** In the second stage we process a half binary tree with modulus switching on each device. We need 3 kernels per polynomial per CRT element for each of NTT and INTT, 1 kernel per polynomial for coefficient wise multiplication in NTT domain and 1 kernel per polynomial for modulus switching. A polynomial reduction is performed after multiplication. We hide polynomial reduction at the beginning of modulus switching and inverse-CRT, instead of an extra kernel only for polynomial reduction.

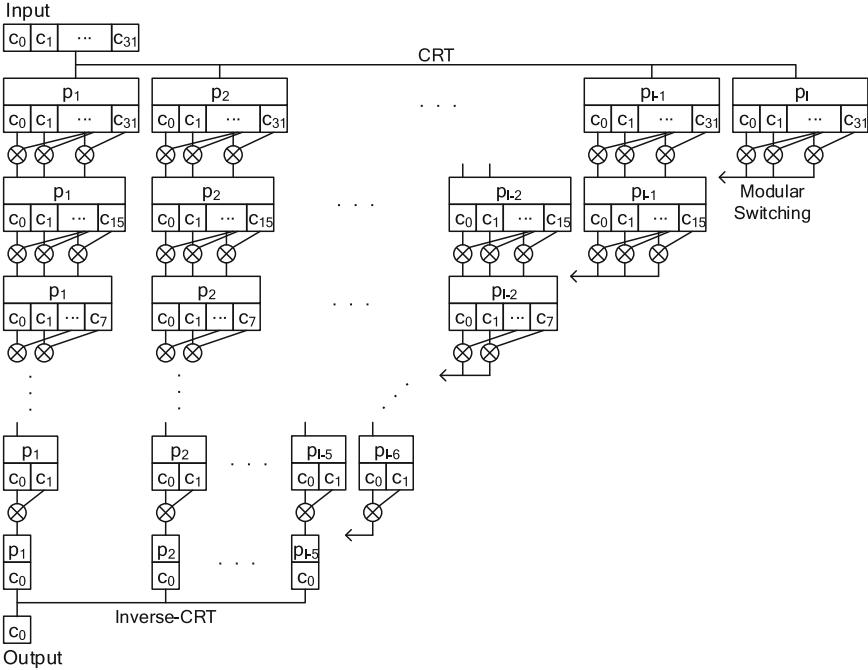


Fig. 1. Realization of the comparison circuit in the PIR scheme using single GPU for database with 2^{32} entries.

4 Performance

We implemented the DSH-PIR with both the Single and Bundled Querying modes using Nvidia GeForce GTX 690¹ running @915 Mhz with 3072 stream processors and 4 GBytes of memory. In Table 1, we compare our query/response sizes with DSH scheme [18] for different entry sizes $N = 2^{2^d}$. In our implementation query/response sizes are slightly larger because we choose the closest modulus in DSH scheme that holds $q = 24^k$. In Table 2, we compare our Bundled/Single Query computation times with DSH implementation which the timings are normalized with message slot size ε . For an index comparison of a Single Query, we achieved 15 times speedup for $d = 5$ and ~ 33 times for $d = \{4, 3\}$ cases. In data aggregation², we achieved 4–6 times speedup compared to DSH Scheme. In Table 3, we compare the Query Size of our scheme with BGN, O-K and DSH schemes for various database sizes. Our ciphertext sizes are (almost) identical to those of the DSH scheme. When compared to the

¹ The NVIDIA GeForce GTX 690 series actually consist of two GTX 680 series graphical processors.

² In cases where we extract entries with more than 1-bit size, we use the same index comparison result to process the remaining bits of a database entry. Also timings that given on the table are per polynomial operation and they are not normalized.

Table 1. Polynomial parameters and Query/Response sizes necessary to support various database sizes N .

Schemes	N	$(\log q, n)$	ε	Query size (MB)	Response size (KB)
DSH [18]	2^{32}	(512, 16384)	1024	32	784
	2^{16}	(250, 8190)	630	3.9	154
	2^8	(160, 4096)	256	0.625	44
Ours	2^{32}	(528, 16384)	1024	33	796
	2^{16}	(264, 8190)	630	4.12	164
	2^8	(168, 4096)	256	0.656	46.8

Table 2. Index comparison and data aggregation times per entry in the database for (d, ε) choices of $(5, 1024)$, $(4, 630)$ and $(3, 256)$ on GPU.

	Depth (d)	Bundled query (msec)			Single query (msec)		
		5	4	3	5	4	3
DSH [18]	Index comparison	4.45	0.71	0.31	4.56	2.03	1.29
	Data aggregation	0.22	0.09	0.04	37	7.45	3.40
Ours	Index comparison	0.26	0.04	0.02	0.31	0.06	0.04
	Data aggregation	0.037	0.005	0.004	9.60	1.26	0.71
Speedup	Index comparison	$\times 17$	$\times 18$	$\times 15$	$\times 15$	$\times 34$	$\times 32$
	Data aggregation	$\times 6$	$\times 18$	$\times 10$	$\times 4$	$\times 6$	$\times 5$

Table 3. Comparison of query sizes for databases upto 2^{32} , 2^{16} and 2^8 entries. Bandwidth complexity is given in the number of ciphertexts; α denotes the ciphertext size.

	BW compl.	α	Query size				
			$d = 5$	$d = 4$	$d = 3$		
Boneh-Goh-Nissim	$\alpha\sqrt{N}$	6144	6144	6144	96 MB	384 KB	24 KB
Kushilevitz-Ostrovsky	$\alpha\sqrt{N}$	2048	2048	2048	32 MB	128 KB	8 KB
DSH [18] (Single)	$\alpha \log N$	1 MB	249 KB	80 KB	32 MB	3.9 MB	640 KB
DSH [18] (Bundled)	$\alpha \log N$	1 KB	406 B	130 B	32 KB	6.32 KB	2.5 KB
Ours (Single)	$\alpha \log N$	1.03 MB	263 KB	84 KB	33 MB	4.1 MB	672 KB
Ours (Bundled)	$\alpha \log N$	1.03 KB	429 B	336 B	33 KB	6.34 KB	2.6 KB

K-O scheme in Bundled Query mode, our ciphertext sizes are three orders of magnitude smaller for $d = 5$ and an order of magnitude smaller for $d = \{4, 3\}$.

In Table 4, we compare our timing and ciphertext size estimates for a real time application given by Aguilar-Melchor and Gaborit [9]. The information given in the table is for $N = 1024$ entries with each entry holding 3 MBytes of data. We select $d = 4$ and assume both GPUs are running data aggregation tasks. In bundled query mode, we are better both in query size and amortized timings compared to other schemes results with the exception for query size of Gentry and Ramzan [6].

Table 4. Comparison of various schemes for real time applications.

Scheme	Query size	Computation time
Lipmaa	2 Mb	33 h
Gentry and Ramzan	3 Kb	17 h
Aguilar-Melchor and Gaborit	300 Mb	10 min
Ours (Single)	20.6 Mb	8.8 h
Ours (Bundled)	33.4 Kb	1.5 min

Acknowledgments. Funding for this research was in part provided by the US National Science Foundation CNS Award #1319130.

References

1. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* **45**, 965–981 (1998)
2. Chor, B., Gilboa, N.: Computationally private information retrieval (extended abstract). In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 304–313. ACM, New York (1997)
3. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract) (1996)
4. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. *FOCS* **1997**, 364–373 (1997)
5. Cachin, C., Micali, S., Stadler, M.A.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) *EUROCRYPT* 1999. LNCS, vol. 1592, p. 402. Springer, Heidelberg (1999)
6. 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)
7. 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)
8. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) *TCC* 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
9. Aguilar-Melchor, C., Gaborit, P.: A lattice-based computationally-efficient private information retrieval protocol (2007)
10. 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)
11. Gentry, C., Halevi, S.: Implementing gentry’s fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) *EUROCRYPT* 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd ITCS, ITCS 2012, pp. 309–325. ACM, New York (2012)

13. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
14. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013)
15. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Annual ACM STOC, STOC 2012, pp. 1219–1234. ACM New York (2012)
16. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
17. Smart, N., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Crypt. **71**, 57–81 (2014)
18. Doröz, Y., Sunar, B., Hammouri, G.: Bandwidth efficient PIR from NTRU. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 195–207. Springer, Heidelberg (2014)
19. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
20. Stehlè, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K. (ed.) Advances in Cryptology-EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)
21. Wang, W., Hu, Y., Chen, L., Huang, X., Sunar, B.: Accelerating fully homomorphic encryption using GPU. In: HPEC, IEEE, pp. 1–5 (2012)
22. Dai, W., Doröz, Y., Sunar, B.: Accelerating ntru based homomorphic encryption using gpus. (2014)
23. Schönhage, A., Strassen, V.: Schnelle multiplikation großer zahlen. Computing **7**, 281–292 (1971)
24. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex fourier series. Math. Comput. **19**, 297–301 (1965)
25. Emmart, N., Weems, C.C.: High precision integer multiplication with a gpu using strassen’s algorithm with multiple fft sizes. PPL **21**, 359–375 (2011)
26. Barrett, P.: Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 311–323. Springer, Heidelberg (1987)

Combining Secret Sharing and Garbled Circuits for Efficient Private IEEE 754 Floating-Point Computations

Pille Pullonen^{1,2} and Sander Siim^{1,2(\square)}

¹ Cybernetica AS, Tartu, Estonia

² University of Tartu, Tartu, Estonia

{pille.pullonen,sander.siim}@cyber.ee

Abstract. Two of the major branches in secure multi-party computation research are secret sharing and garbled circuits. This work succeeds in combining these to enable seamlessly switching to the technique more efficient for the required functionality. As an example, we add garbled circuits based IEEE 754 floating-point numbers to a secret sharing environment achieving very high efficiency and the first, to our knowledge, fully IEEE 754 compliant secure floating-point implementation.

1 Introduction

Secure multi-party computation (MPC) enables parties to securely compute some function on their secret inputs and receive the secret outputs, without leaking anything to other parties. The fastest MPC protocols for integer arithmetic, like Sharemind [5, 8] and SPDZ [10], rely on additive secret sharing. Additive sharing supports efficient addition and multiplication due to the algebraic properties of the scheme. However, floating-point arithmetic is much more sophisticated and contains a composition of different operations, both integer arithmetic as well as bitwise operations. Existing implementations based on secret sharing provide near approximations to the IEEE 754 standard [1, 17, 22]. Although [11] proposes IEEE 754 protocols, no implementation is provided.

Another MPC approach is based on the garbled circuits method (GC) attributed to Yao [29] and detailed in [21]. A good overview of the recent advances can be found in [2, 3], especially in the full versions. The baseline method is applicable to the two-party setting, however it can be extended to the case with more parties [4]. State-of-the-art garbling methods are already very efficient [2] and, in addition, means to derive optimized circuits from existing programs have been developed [14, 19]. This allows secure protocols for arbitrary computations to be built with small effort using a general GC approach.

This research was, in part, funded by the U.S. Government. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. This work has also received funding from the Estonian Research Council through grant IUT27-1, and ERDF through EXCS.

However, in many cases the obtained protocols are less efficient than their secret-sharing-based alternatives. In practice, it would be useful to choose the more efficient technique, either secret sharing or GC, for each particular subprotocol, but this requires interleaving secret sharing and GC based protocols in one computation.

In this paper, we present a *hybrid protocol*, which enables arbitrary secure computations through a combination of GC and secret sharing protocols. In our protocol, GC gives the power to do bit-level operations in a compact manner, whereas secret sharing complements the construction with a fast oblivious transfer as well as composability with other secret-sharing-based protocols. Thereby, large and complex algorithms can be implemented by composition of the most efficient basic primitives. We illustrate the benefits by extending the Sharemind MPC framework [5,8] with, to our knowledge, the first secure floating-point protocol suite fully conforming to the IEEE 754 standard.

2 Preliminaries

In MPC, parties $\mathcal{P}_1, \dots, \mathcal{P}_m$ want to securely compute a function f on secret inputs x_1, \dots, x_m to learn $f(x_1, \dots, x_m) = (y_1, \dots, y_m)$, without leaking anything about inputs x_i to other parties. Secret sharing is a mechanism of distributing data between participants without giving any of them direct access to the data, but enabling computations [27]. We denote a secret-shared vector \mathbf{x} of n elements shared between parties $\mathcal{P}_1, \dots, \mathcal{P}_m$ by $[\![\mathbf{x}]\!] = [\![x_1, x_2, \dots, x_n]\!] = [\![x_1]\!], \dots, [\![x_n]\!]$ where party \mathcal{P}_i holds $[\![\mathbf{x}]\!]_i = [\![x_1, x_2, \dots, x_n]\!]_i$. We focus on the additive secret sharing scheme, where sharing is defined with $\sum_{i=1}^m [\![\mathbf{x}]\!]_i = x$. However, other schemes may also be used to implement our proposed protocol.

In the garbled circuits protocol [21, 29], two parties called garbler and evaluator securely compute a known function $f(x, y)$ on their joint inputs. The garbler encrypts a Boolean circuit of $g = f(a, \cdot)$ and sends the *garbled* truth tables of gates to the evaluator. The evaluator then uses oblivious transfer to obtain the keys corresponding to its input to decrypt the garbled circuit and evaluate $g(b)$.

We use notation from [2] to describe circuits as a tuple $f = (n, m, q, A, B, G)$, where n, m and q respectively denote the number of external input wires, external output wires and gates in f . All wires are labelled by indexes. Namely, 1 to n are input wires, $n + 1$ to $n + q$ mark gate output wires and $n + q - m + 1$ to $n + q$ are circuit outputs. Functions A and B , respectively, identify the first and second input wire of any gate. For each gate g in f , the function $G(g) : \{0, 1\}^2 \rightarrow \{0, 1\}$ denotes the functionality of g . We use $X_j^b \in \{0, 1\}^k$ to denote the token of the j -th wire corresponding to bit $b \in \{0, 1\}$, where k is the length of the generated tokens. We say X_j^b has the *semantics* of b and type $\text{1sb}(X_j^b)$.

Here we only emphasize the important aspects of the used security proof framework, for details we refer to [6]. A protocol is said to be *input private* if, for any collection of allowed corrupted parties, there exists a simulator that can simulate the view of the adversary based on the inputs of corrupted parties. The *ordered composition* of an input private and a secure protocol, where all outputs

are provided by the secure protocol, is secure if it is *output predictable*. The latter means that the composed protocols are correct and the final protocol does not leak information about its input shares to ensure the privacy of the first part.

Garbled circuits have two important security definitions: privacy and obliviousness [3]. Respectively, we consider prv.ind , prv.sim and obv.ind , obv.sim for either indistinguishability or simulation based versions of these definitions. Both properties are formalised via the *side-information function* Φ that captures the information that is revealed by the garbled circuit. We consider Φ_{topo} and Φ_{xor} that leak the topology and XOR operations. These functions are both efficiently invertible [2]. Therefore, by equivalence relations from [3], indistinguishability and simulation-based definitions coincide for both privacy and obliviousness.

3 Combining Garbled Circuits with Secret Sharing

Our goal is to construct an efficient protocol for securely evaluating Boolean circuits on bitwise secret-shared input, thereby allowing secret sharing protocols to be composed with computations more suitable for GC. Thus, each subprotocol can use the method more suitable for the given functionality and inputs. A similar approach is also used in the TASTY framework that combines GC and additively homomorphic encryption in a two-party setting [13].

The idea of our protocol is to set up GC to accept secret-shared inputs and to produce shared outputs. Thus, our protocol in Algorithm 1 consists of three important steps. First, we require an efficient oblivious garbling scheme to securely evaluate circuits. Suppose we have $\mathcal{CP}_1, \dots, \mathcal{CP}_m$ who hold some secret-shared data. We will let computing parties \mathcal{CP}_1 and \mathcal{CP}_2 respectively perform the computations of garbler and evaluator from the GC protocol. Note that additionally to the properties of the secret sharing scheme we require that \mathcal{CP}_1 and \mathcal{CP}_2 are not colluding. Second, the GC protocol requires a special oblivious transfer (OT) to provide the input tokens to the evaluator from the secret-shared inputs. Third, we must convert the garbled outputs to the appropriate secret-shared form.

Algorithm 1. Hybrid protocol for processing bitwise secret-shared data with a garbled circuit

Input: Shared bit vector $\llbracket \mathbf{x} \rrbracket = \llbracket x_1, \dots, x_n \rrbracket$
 Boolean circuit f that calculates $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$

Output: Shared bit vector $\llbracket \mathbf{y} \rrbracket = \llbracket y_1, \dots, y_m \rrbracket$ such that $\llbracket \mathbf{y} \rrbracket = f(\llbracket \mathbf{x} \rrbracket)$

- 1 **foreach** input wire $i \in \{1, \dots, n\}$ **do**
- 2 \mathcal{CP}_1 generates a token pair $(X_i^0, X_i^1) \in \{0, 1\}^k \times \{0, 1\}^k$
- 3 The computing parties initiate an OT protocol which results in \mathcal{CP}_2 receiving $\mathbf{X} = \{X_1^{x_1}, \dots, X_n^{x_n}\}$ (the input tokens corresponding to the actual input bits)
- 4 \mathcal{CP}_1 garbles circuit f and sends the garbled truth tables to \mathcal{CP}_2
- 5 \mathcal{CP}_2 evaluates garbled f using \mathbf{X} to get output tokens $\mathbf{Y} = \{X_{o_1}^{y_1}, \dots, X_{o_m}^{y_m}\}$
- 6 The garbled output is converted to secret-shared form to receive $\llbracket \mathbf{y} \rrbracket$
- 7 **return** $\llbracket \mathbf{y} \rrbracket$

Algorithm 2. Oblivious transfer of input tokens (**OT**)

Input: \mathcal{CP}_1 holds the input tokens $\{X_1^0, \dots, X_n^0, X_1^1, \dots, X_n^1\}$
The input bit vector $[\mathbf{x}] = [x_1, \dots, x_n]$ is shared between all parties
Output: \mathcal{CP}_2 receives input tokens $\{X_1^{x_1}, \dots, X_n^{x_n}\}$

- 1 $[\mathbf{X}^0] = [X_1^0, \dots, X_n^0]$ and $[\mathbf{X}^1] = [X_1^1, \dots, X_n^1]$ are instantiated as shared values, with shares of \mathcal{CP}_2 and \mathcal{CP}_3 initialized to 0
- 2 $[\mathbf{X}] \leftarrow [\mathbf{X}^0] \cdot ([\mathbf{1}] - [\mathbf{x}]) + [\mathbf{X}^1] \cdot [\mathbf{x}]$
- 3 \mathcal{CP}_1 and \mathcal{CP}_3 send their shares of $[\mathbf{X}]$ to \mathcal{CP}_2
- 4 \mathcal{CP}_2 combines the shares of $[\mathbf{X}]$ to get $\{X_1^{x_1}, \dots, X_n^{x_n}\}$
- 5 **return** $\{X_1^{x_1}, \dots, X_n^{x_n}\}$

3.1 An Implementation of the Hybrid Protocol

Generally, the hybrid protocol can be implemented using various secret sharing and garbling schemes, provided they retain the security properties from Sect. 3.2 and conversion protocols in Algorithm 1 exist. However, we will focus on our instantiation built into the Sharemind MPC platform [5]. We chose Sharemind because it already provides an optimized multi-party computation environment based on secret sharing, which could easily be extended with our GC based protocol.

Our protocol extends Sharemind’s `additive3pp` protection domain, which implements various secure computation protocols using 3-out-of-3 additive secret sharing [7]. This allows us to easily compose the hybrid protocol with fast existing primitives for integer arithmetic. Note that different data types provided by Sharemind can be efficiently converted to shared bit vectors required in our construction [8]. Consequently, we fix a setting with three computing parties \mathcal{CP}_1 , \mathcal{CP}_2 and \mathcal{CP}_3 and additive secret sharing. As with `additive3pp` protocols, our construction provides security against a single passively corrupted party.

Oblivious Transfer. The garbler \mathcal{CP}_1 generates a pair (X_i^0, X_i^1) of tokens for every bit x_i in the beginning of the garbling process. We need to transfer the tokens that correspond to the protocol inputs to the evaluator \mathcal{CP}_2 . Clearly, if we have a subprotocol that calculates the necessary secret-shared tokens $[X_1^{x_1}, \dots, X_n^{x_n}]$, then we can complete the transfer by sending all result shares to \mathcal{CP}_2 . This subprotocol can be easily implemented using secret-sharing-based multiplication and addition protocols. The resulting **OT** protocol is given in Algorithm 2. In addition to basic OT security properties, we also require that the inputs x_i are not leaked.

The computations on line 2 are performed using the secure and input private multiplication and addition protocols from [8]. As a result, $[\mathbf{X}] = [X_1^{x_1}, \dots, X_n^{x_n}]$ and the inputs $[\mathbf{x}]$ remain private. Note that each x_i can easily be extended to the length of the tokens, therefore the operations are performed in \mathbb{Z}_{2^k} . On line 3, the shares $[\mathbf{X}]$ are sent to \mathcal{CP}_2 who combines them to receive the input tokens.

Garbling. The emphasis of efficient GC is on reducing network communication, as this is the bottleneck for GC protocols. Recent garbling schemes bring the cost

of local computations to a minimum as demonstrated in [2]. Due to these considerations, we chose the GAXR scheme with the A4 DKC instantiation from [2] for our protocol, which is one of the fastest to date. Other garbling schemes could be used just as well, provided they retain the obliviousness property [3]. The GAXR scheme incorporates *free-XOR* [18] and *garbled row reduction* [24] optimizations, both of which significantly reduce network communication of GC.

The authors of [2,3] formalize the underlying encryption primitive of the garbling process as a *dual-key cipher*. The dual-key cipher used in the GAXR scheme is a function $\mathbb{E} : \{0, 1\}^k \times \{0, 1\}^k \times \{0, 1\}^\tau \times \{0, 1\}^k \rightarrow \{0, 1\}^k$. It takes secret wire tokens A and B and a tweak T to encrypt a wire token X , resulting in a ciphertext $\mathbb{E}(A, B, T, X) = \pi(K \parallel T)_{[1:k]} \oplus K \oplus X$, where $K = 2A \oplus 4B$ and $X, A, B \in \{0, 1\}^k$ and $T \in \{0, 1\}^\tau$. Here $\pi(K \parallel T)_{[1:k]}$ denotes the first k bits of the result. The decryption is completely symmetric.

The function $\pi : \{0, 1\}^{k+\tau} \rightarrow \{0, 1\}^{k+\tau}$ denotes a random permutation, as the security of GAXR is shown in the *random permutation model*. We use a fixed-key AES-128 with $k = 80$ and $\tau = 48$ to instantiate π , which provides reasonable security guarantees for this garbling scheme [2]. Tweak T is the encrypted gate index encoded as a τ -bit integer. For the doubling function denoted by $2A$ we use multiplication with element x over finite field $GF(2^k)$, as it provides the best security guarantees over other possible alternatives [2]. Here k corresponds to bit-length of the wire tokens. Our implementation uses the irreducible polynomial $x^{80} + x^9 + x^4 + x^2 + 1$ from [26] for defining the finite field.

Figure 1 summarizes the hybrid protocol. The garbler \mathcal{CP}_1 first generates a token pair (X_i^0, X_i^1) for each input wire, with X_i^0 and X_i^1 having the semantics of 0 and 1 respectively. Then all three computing parties synchronously execute the OT protocol in Algorithm 2. As a result \mathcal{CP}_2 receives the correct input tokens needed for evaluation. Next, \mathcal{CP}_1 garbles the circuit according to the GAXR scheme and sends the garbled truth tables P to \mathcal{CP}_2 . The evaluator \mathcal{CP}_2 can then evaluate the garbled circuit using the transferred input tokens to receive the garbled output.

As an implementation detail, we have parallelized our protocol on two levels. First, the garbled tables are streamed by fixed-size batches from garbler to evaluator, similarly to [15]. The evaluator can then start evaluating the circuit while the garbler encrypts the next batch. This is especially relevant performance-wise for large circuits. The batch size can be fixed for different circuits separately and fine-tuned to match the Sharemind instance's network and hardware capabilities.

In addition, our implementation allows both garbler and evaluator to run several threads to evaluate the same circuit with different inputs simultaneously. This can be thought of as using a number of garbler-evaluator pairs, similarly to the *cut-and-choose* implementation of [20] for actively secure GC. Besides parallel garbling, this allows a joint OT to be done for all the scheduled evaluations. This parallelization greatly reduces the cost of a single circuit evaluation.

Resharing. The final step in the protocol is resharing the output between all three computing parties using perfectly secure **Reshare** protocol Algorithm 1 from [5]. This protocol rerandomizes the output shares held by \mathcal{CP}_1 and \mathcal{CP}_2 as

Algorithm 3: Hybrid protocol algorithm for \mathcal{CP}_1 <pre> Input: $\llbracket \mathbf{x} \rrbracket_1 = \llbracket x_1, \dots, x_n \rrbracket_1$ and circuit $f = (n, m, q, A, B, G)$ Output: $\llbracket \mathbf{y} \rrbracket_1 = \llbracket y_1, \dots, y_m \rrbracket_1$ such that $\llbracket \mathbf{y} \rrbracket = f(\llbracket \mathbf{x} \rrbracket)$ $R \xleftarrow{\\$} \{0, 1\}^{k-1} \ 1$ for $i \leftarrow 1$ to n do $t \xleftarrow{\\$} \{0, 1\}$ $X_i^0 \xleftarrow{\\$} \{0, 1\}^{k-1} \ t, X_i^1 \leftarrow X_i^0 \oplus R$ OT $([X_1^0, \dots, X_n^0], [X_1^1, \dots, X_n^1], \llbracket \mathbf{x} \rrbracket_1)$ for $g \leftarrow n+1$ to $n+q$ do $a \leftarrow A(g), b \leftarrow B(g)$ if $G(g) = XOR$ then $X_g^0 \leftarrow X_a^0 \oplus X_b^0, X_g^1 \leftarrow X_g^0 \oplus R$ else for $i \leftarrow 0$ to $1, j \leftarrow 0$ to 1 do $u \leftarrow i \oplus \text{lsb}(X_a^0)$ $v \leftarrow j \oplus \text{lsb}(X_b^0)$ $r \leftarrow G(g, u, v)$ if $i = 0$ and $j = 0$ then $X_g^r \leftarrow \mathbb{E}(X_a^u, X_b^v, g, 0^k)$ $X_g^{r-1} \leftarrow X_g^r \oplus R$ else $P[g, i, j] \leftarrow \mathbb{E}(X_a^u, X_b^v, g, X_g^r)$ Send P to \mathcal{CP}_2 for $i \leftarrow 1$ to m do $y'_{i1} \leftarrow \text{lsb}(X_{n+q-m+i}^0)$ $\llbracket \mathbf{y} \rrbracket'_1 \leftarrow [y'_{11}, \dots, y'_{m1}]$ $\llbracket y_1, \dots, y_m \rrbracket_1 \leftarrow \text{Reshare}(\llbracket \mathbf{y} \rrbracket'_1)$ return $\llbracket \mathbf{y} \rrbracket_1 = \llbracket y_1, \dots, y_m \rrbracket_1$ </pre>	Algorithm 4: Hybrid protocol algorithm for \mathcal{CP}_2 <pre> Input: $\llbracket \mathbf{x} \rrbracket_2 = \llbracket x_1, \dots, x_n \rrbracket_2$ and circuit $f = (n, m, q, A, B, G)$ Output: $\llbracket \mathbf{y} \rrbracket_2 = \llbracket y_1, \dots, y_m \rrbracket_2$ such that $\llbracket \mathbf{y} \rrbracket = f(\llbracket \mathbf{x} \rrbracket)$ $[X_1, \dots, X_n] \leftarrow \text{OT}(0^{k-n}, 0^{k-n}, \llbracket \mathbf{x} \rrbracket_2)$ Receive P from \mathcal{CP}_1 for $g \leftarrow n+1$ to $n+q$ do $a \leftarrow A(g), b \leftarrow B(g)$ $i \leftarrow \text{lsb}(X_a), j \leftarrow \text{lsb}(X_b)$ if $G(g) = XOR$ then $X_g \leftarrow X_a \oplus X_b$ else if $i = 0$ and $j = 0$ then $X_g \leftarrow \mathbb{E}(X_a, X_b, g, 0^k)$ else $X_g \leftarrow \mathbb{D}(X_a, X_b, g, P[g, i, j])$ for $i \leftarrow 1$ to m do $y'_{i2} \leftarrow \text{lsb}(X_{n+q-m+i})$ $\llbracket \mathbf{y} \rrbracket'_2 \leftarrow [y'_{12}, \dots, y'_{m2}]$ $\llbracket y_1, \dots, y_m \rrbracket_2 \leftarrow \text{Reshare}(\llbracket \mathbf{y} \rrbracket'_2)$ return $\llbracket \mathbf{y} \rrbracket_2 = \llbracket y_1, \dots, y_m \rrbracket_2$ </pre>
Algorithm 5: Hybrid protocol algorithm for \mathcal{CP}_3 <pre> Input: $\llbracket \mathbf{x} \rrbracket_3 = \llbracket x_1, \dots, x_n \rrbracket_3$ Output: $\llbracket \mathbf{y} \rrbracket_3 = \llbracket y_1, \dots, y_m \rrbracket_3$ such that $\llbracket \mathbf{y} \rrbracket = f(\llbracket \mathbf{x} \rrbracket)$ OT $(0^{k-n}, 0^{k-n}, \llbracket \mathbf{x} \rrbracket_3)$ $\llbracket \mathbf{y} \rrbracket'_3 \leftarrow 0^m$ $\llbracket y_1, \dots, y_m \rrbracket_3 \leftarrow \text{Reshare}(\llbracket \mathbf{y} \rrbracket'_3)$ return $\llbracket \mathbf{y} \rrbracket_3 = \llbracket y_1, \dots, y_m \rrbracket_3$ </pre>	

Fig. 1. Detailed algorithms of the hybrid protocol for all computing parties.

$\mathbf{y} = \llbracket \mathbf{y} \rrbracket'_1 + \llbracket \mathbf{y} \rrbracket'_2$ to a uniformly secret-shared output $\llbracket \mathbf{y} \rrbracket$ and ensures that we can securely compose our protocol with all **additive3pp** protocols, which is vital for efficient computations that would benefit from both GC and secret sharing.

3.2 Security of the Hybrid Protocol

Based on [6], we need to prove that the protocol up until the **Reshare** function is passively input private and then apply the composition result from [6]. For this, we also need to establish the output predictability of the composition. We denote the part of the hybrid protocol on Fig. 1 before final **Reshare** protocol as **Hybrid'**. This section gives an overview of the important aspects of the proof, a full proof can be found in the full version of this paper [25]. Note, that the

obliviousness of the garbling scheme [3] is quite like the input privacy [6] and is necessary for the input privacy of the **Hybrid'** protocol.

Theorem 1. *Ordered composition of **Hybrid'** and **Reshare** is output predictable.*

Proof (Proof sketch). Clearly, **Hybrid'** and **Reshare** are in ordered composition because all outputs of **Hybrid'** are inputs to **Reshare**. There is no data flow from **Reshare** to **Hybrid'**. The correctness of **Hybrid'** follows from the correctness of the sub-protocols used in the OT part and the correctness of the GAXR garbling scheme. Therefore, output predictability follows from Lemma 2 in [6].

Theorem 2. *GAXR scheme is computationally obv.ind and obv.sim secure.*

Proof (Proof sketch). The types of the input wires are independent of the semantics as they are generated independently on line 3 by \mathcal{CP}_1 . In short, the obv.ind security follows from the fact that the keys of the outputs are generated the same way as the intermediate keys. Therefore, if there exists an adversary that breaks the obv.ind security for two functions f_1 and f_2 then this adversary can be extended to break the prv.ind security for two functions $c \circ f_1$ and $c \circ f_2$ for a constant function c . Finally, obv.ind security and obv.sim security coincide.

Theorem 3. *Protocol **Hybrid'** is perfectly input private for statically corrupted \mathcal{CP}_1 or \mathcal{CP}_3 and computationally input private against corrupted \mathcal{CP}_2 .*

Proof (Proof sketch). We have to show the existence of the privacy simulator that can simulate the view of the corrupted party based on its inputs.

Corrupted \mathcal{CP}_1 or \mathcal{CP}_3 . The only incoming communication for these protocols occurs during the OT computation phase. Therefore, the perfect input privacy of these parties is ensured by the perfect input privacy of the addition and multiplication protocol and the composability of input privacy (Theorem 3 in [6]).

Corrupted \mathcal{CP}_2 . From obv.sim security in Theorem 2 we know that there exists a simulator \mathcal{S} such that, for inputs $\mathcal{S}(1^k, \Phi(f))$, it outputs (F, X) indistinguishable from those output by the garbling scheme. This simulator is defined by the game ObvSim in [3]. The privacy simulator \mathcal{P} for \mathcal{CP}_2 can be built from the simulator \mathcal{S} . This \mathcal{P} knows the circuit f and also has the security parameter k , therefore, it can run $\mathcal{S}(1^k, \Phi(f))$ to obtain (F, X) . Next, it has to simulate the OT that can be done perfectly by using the privacy simulator for the computation part and simulating the declassifying procedure with output X . All of the simulation, except for the choice of F and X , is perfect. Therefore, if the adversary gains any power to distinguish between the real life and simulation \mathcal{P} , it must result from the values F and X . However, this would invalidate the obv.sim security.

Corollary 4. *Hybrid protocol (Fig. 1) is perfectly secure against passively corrupted \mathcal{CP}_1 and \mathcal{CP}_3 and computationally secure against passively corrupted \mathcal{CP}_2 .*

Proof. The composition is jointly output predictable (Theorem 1) and **Hybrid'**, first part of the ordered composition, is input private (Theorem 3). Using the composition result (Theorem 2 in [6]) we conclude that the full hybrid protocol is secure.

4 Using the Hybrid Protocol for Efficient Computations

Sharemind’s `additive3pp` protocols enable fast integer operations. On the other hand, bit-level operations are more costly. However, in practical applications we are also interested in more complex primitives that rely heavily on bit-level operations. A very relevant example of this is floating-point computations.

The *de facto* standard today for binary floating-point arithmetic is IEEE 754 [16]. Although existing secure implementations of floating-point operations resemble IEEE 754 [1, 17, 22], they do not always produce identical results compared to regular hardware implementations. The main shortcomings are in not rounding inexact results to nearest representable floating-point numbers, lack of support for gradual underflow and missing error handling [12].

Using the CBMC-GC circuit compiler [14] (v.0.9.3 [9]), we were able to implement an efficient and fully IEEE 754 compliant floating-point protocol suite based on our hybrid protocol. The CBMC-GC compiler transforms C programs directly to highly optimized circuits usable in a GC protocol. This allowed us to use exact IEEE 754 software implementations as a basis for our protocols.

We implemented both single and double precision secret-shared floating-point data types. The `float` and `double` types are represented as 32-bit and 64-bit bitwise secret-shared integers that correspond exactly to the IEEE 754 standard. Our construction guarantees bit-by-bit identical results to those of regular hardware floating-point procedures, excluding non-standardized details such as the significand bits of a NaN. We empirically verified this claim for the four arithmetic operations and square root on a machine with Intel Core i7-870 2.93 GHz processor against equivalent C programs compiled with GCC 4.8.1-2.

4.1 Circuits for IEEE 754 Primitives

The circuits used in our protocol suite are listed and described in Table 1. We list circuit sizes as well as the number of garbled tables batches sent during one evaluation of the circuit. The circuits were compiled on a workstation with 16 GB RAM and an Intel Core i7-870 2.93 GHz processor. Although it would have much reduced the circuit sizes, we were unable to use the SAT-minimization functionality of CBMC-GC for larger circuits due to high compilation times.

We used the efficient SoftFloat [28] IEEE 754 software implementation for compiling addition, multiplication, division and square root circuits. We additionally used musl libc [23] for double precision e^x and error function (erf) as an example of more complex operations and the flexibility of our approach to implement arbitrary primitives. Only minor syntactic modifications of the source code were required to compile it with CBMC-GC. We hardcoded rounding to the default “Round to nearest even” mode defined in IEEE 754, since this is most used in practice and provides the best bounds on rounding errors [12]. Alternatively, we could give the rounding mode as input to the circuit.

We chose to ignore all floating-point exceptions that may be raised during computations, since in an MPC environment, raising an exception (e.g. division by zero) in the middle of a computation can possibly leak information about

Table 1. IEEE 754 floating-point operation circuits compiled with CBMC-GC

Circuit	Non-XOR gates	Total gates	No of batches	Used SAT-minimization	Compilation time
float_add	5671	7052	1	+	8 min 32 s
float_sub	5671	7052	1	+	5 min 28 s
float_mul	5138	7701	1	+	5 min 1 s
float_div	12851	21384	1	-	58 s
float_sqrt	35987	66003	2	-	2 min 40 s
double_add	13129	15882	1	+	1 h 8 min
double_sub	13129	15882	1	+	1 h 11 min
double_mul	13104	25276	1	+	3 h 46 min
double_div	36133	73684	2	-	5 min 35 s
double_sqrt	85975	169932	4	-	10 min 11 s
double_exp	393807	579281	8	-	1 h 13 min
double_erf	2585188	3979603	52	-	47 h 4 min

inputs. As our protocols correctly handle all special cases defined in the standard (NaNs, infinities, denormalized numbers), any exceptions will be reflected in the final result. Previous implementations [1, 17, 22] did not explicitly handle such cases and produced valid but meaningless results in error situations.

4.2 Performance Analysis

We benchmarked the performance of our implemented IEEE 754 primitives as well as the existing floating-point operations [17] in Sharemind for comparison. The benchmarks were performed on a cluster of three nodes hosting Sharemind. All nodes had 48 GB of RAM and a 12-core 3 GHz Intel CPU supporting AES-NI and HyperThreading. The nodes were connected to a LAN with 1 Gbps full duplex links. All tests were executed with a maximum of 24 concurrent garbler-evaluator pairs, as the hardware supports up to 24 parallel threads.

The performance results are shown in Table 2 for single precision and Table 3 for double precision. All measurements are presented in operations per second (ops) as the mean of 5 to 1000 iterations depending on the circuit size. The measurements depict the whole running time of the protocol including oblivious transfer, garbling and evaluation. Circuits are parsed and cached in an offline phase, however. The input size refers to the number of respective operations computed in one test using the parallelization techniques described in Sect. 3.1.

Our measurements show that hybrid protocol IEEE 754 operations, excluding error function, are faster than approximation-based operations for smaller input sizes. The error function clearly illustrates the substantial overhead for evaluating very large circuits, thereby motivating the composition of small but efficient primitives as opposed to full circuit programs. The IEEE 754 division and square root perform very well compared to approximation-based versions, whereas the error function and multiplication are slower on larger input sizes. Our protocols

Table 2. Performance of single precision floating-point operations (ops)

		Input size in elements				
		1	10	100	1000	10000
Add	Approx.	2.43	24.1	228.1	1496	3790
	IEEE 754	24.99	134.5	477.2	583.6	597
Multiply	Approx.	7.76	77.94	751.6	5413	16830
	IEEE 754	26.17	135.5	506	632.9	632.9
Divide	Approx.	0.53	5.25	46.48	237	432.6
	IEEE 754	14.53	88.2	233.6	279.1	284.5
Square root	Approx.	0.34	3.26	28.07	126.1	206.1
	IEEE 754	7.83	44	92.9	105.1	106.6

Table 3. Performance of double precision floating-point operations (ops)

		Input size in elements			
		1	10	100	1000
Add	Approx.	2.29	22.22	188.2	857.7
	IEEE 754	16	103	228	260
Multiply	Approx.	7.17	71.98	647.9	3560
	IEEE 754	13.74	90.8	221	259
Divide	Approx.	0.5	4.78	35.22	115.7
	IEEE 754	7.31	46	89.2	101
Square root	Approx.	0.26	2.4	14.23	31
	IEEE 754	3.57	23.3	39.5	43.4
e^x	Approx.	0.28	2.57	14.7	31.1
	IEEE 754	1.1	6.38	9	9.5
Error function	Approx.	0.3	2.92	19.8	55.4
	IEEE 754	0.18	0.95	1.35	1.47

well outperform the results from [22] and our double precision addition and division are faster than the implementation of [1], however, multiplication is slightly slower. The latter is expected, since it is efficient to implement floating-point multiplication using secret sharing and less is gained from a GC approach.

The results also show that IEEE 754 operations do not benefit much from parallelization already after inputs of size ~ 100 , while the approximation-based operations parallelize well to 10000 elements. This is due to the large size of the garbled tables that are transmitted over the network. In all larger tests with the IEEE 754 operations, the network link was constantly saturated, which introduced an inevitable upper bound on performance. This demonstrates the tradeoff between GC and secret sharing, as GC generally requires more network communication, but has better round-complexity. For example, in our instantiation,

the garbled circuit for `float` addition has size \sim 175 KB, whereas the approximation-based protocol uses at most 12 KB of one-way network communication over a series of communication rounds. Consequently, the sharing-based protocols have better amortized performance for larger inputs. In practice, the input size can be used to dynamically choose between GC or sharing-based protocols.

The IEEE 754 protocols used significantly more memory and processing power, as all processor cores of the garbler node were nearly constantly working at maximum capacity. The effect of the high-speed parallel garbling on overall performance was nevertheless ultimately dominated by the network bandwidth, suggesting that less powerful hardware could be used for similar results. The approximation-based counterparts used only \sim 10 % of the hardware capability.

5 Conclusion

This work provided a protocol for combining GC with secret sharing. For this we consider a setting where the oblivious transfer for the garbled evaluation inputs can work for secret-shared inputs rather than the inputs known to the evaluator. In addition, it is required that the outputs of the garbled evaluation remain private. This allows us to combine the strengths of both approaches. Especially, efficient secret-sharing-based computation protocols can be augmented with easily generated GC based protocols for the functionalities where no known efficient sharing-based protocol exists. As an example, we added a very efficient first fully IEEE 754 compliant secure floating-point implementation to Sharemind.

Acknowledgments. We would like to thank the authors of the CBMC-GC circuit compiler for supporting us in our efforts to generate the described circuits.

References

1. Aliasgari, M., Blanton, M., Zhang, Y., Steele, A.: Secure computation on floating point numbers. In: Proceedings of NDSS 2013. The Internet Society (2013)
2. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: Proceedings of SP 2013, pp. 478–492. IEEE Computer Society, Washington, DC (2013)
3. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of CCS 2012, pp. 784–796. ACM, New York (2012)
4. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for secure multi-party computation. In: Proceedings of CCS 2008, pp. 257–266. ACM (2008)
5. Bogdanov, D.: Sharemind: programmable secure computations with practical applications. Ph.D. thesis. University of Tartu (2013)
6. Bogdanov, D., Laud, P., Laur, S., Pullonen, P.: From input private to universally composable secure multi-party computation. In: Proceedings of CSF 2014. IEEE Computer Society (2014)
7. Bogdanov, D., Laud, P., Randmets, J.: Domain-polymorphic programming of privacy-preserving applications. In: Proceedings of PETShop 2013, pp. 23–26. ACM (2013)

8. Bogdanov, D., Niitsoo, M., Toft, T., Willemson, J.: High-performance secure multi-party computation for data mining applications. *IJIS* **11**(6), 403–418 (2012)
9. CBMC-GC. <http://forsyte.at/software/cbmc-gc/>
10. Damgård, I., Pastro, V., Smart, N., 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)
11. Franz, M., Katzenbeisser, S.: Processing encrypted floating point signals. In: Proceedings of MM&Sec 2011, pp. 103–108. ACM, New York (2011)
12. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.* **23**(1), 5–48 (1991)
13. Henecka, W., Kögl, S., Sadeghi, A.R., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: Proceedings of CCS 2010, pp. 451–462. ACM, New York (2010)
14. Holzer, A., Franz, M., Katzenbeisser, S., Veith, H.: Secure two-party computations in ANSI C. In: Proceedings of CCS 2012, pp. 772–783. ACM (2012)
15. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: Proceedings of SEC 2011. USENIX Association (2011)
16. 754-2008 - IEEE standard for floating-point arithmetic (2008). <http://ieeexplore.ieee.org/servlet/opac?punumber=4610933>
17. Kamm, L., Willemson, J.: Secure floating-point arithmetic and private satellite collision analysis. *IJIS* (2014). <http://link.springer.com/article/10.1007%2Fs10207-014-0271-8>
18. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
19. Kreuter, B., Mood, B., Shelat, A., Butler, K.: PCF: a portable circuit format for scalable two-party secure computation. In: Proceedings of SEC 2013, pp. 321–336. USENIX Association, Berkeley (2013)
20. Kreuter, B., Shelat, A., Shen, C.: Billion-gate secure computation with malicious adversaries. In: Proceedings of Security 2012. USENIX Association (2012)
21. Lindell, Y., Pinkas, B.: A proof of security of Yao’s protocol for two-party computation. *J. Cryptol.* **22**(2), 161–188 (2009)
22. Liu, Y.C., Chiang, Y.T., Hsu, T.S., Liau, C.J., Wang, D.W.: Floating point arithmetic protocols for constructing secure data analysis application. *Procedia Comput. Sci.* **22**, 152–161 (2013)
23. musl libc. <http://www.musl-libc.org/>
24. 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)
25. Pullonen, P., Siim, S.: Combining secret sharing and garbled circuits for efficient private IEEE 754 floating-point computations. Cryptology ePrint Archive, Report 2014/990 (2014)
26. Seroussi, G.: Table of low-weight binary irreducible polynomials (1998). <http://www.hpl.hp.com/techreports/98/HPL-98-135.html>
27. Shamir, A.: How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979)
28. SoftFloat. <http://www.jhauser.us/arithmetic/SoftFloat.html>
29. Yao, A.C.: Protocols for secure computations. In: Proceedings of SFCS 1982, pp. 160–164. IEEE Computer Society, Washington, DC (1982)

Cryptanalysis of a (Somewhat) Additively Homomorphic Encryption Scheme Used in PIR

Tancrède Lepoint^{1(✉)} and Mehdi Tibouchi²

¹ CryptoExperts, Paris, France

tancrede.lepoint@cryptoexperts.com

² NTT Secure Platform Laboratories, Tokyo, Japan

tibouchi.mehdi@lab.ntt.co.jp

Abstract. Private Information Retrieval (PIR) protects users' privacy in outsourced storage applications and can be achieved using additively homomorphic encryption schemes. Several PIR schemes with a “real world” level of practicality, both in terms of computational and communication complexity, have been recently studied and implemented. One of the possible building block is a conceptually simple and computationally efficient protocol proposed by Trostle and Parrish at ISC 2010, that relies on an underlying secret-key (somewhat) additively homomorphic encryption scheme, and has been reused in numerous subsequent works in the PIR community (PETS 2012, FC 2013, NDSS 2014, etc.).

In this paper, we show that this encryption scheme is not one-way: we present an attack that decrypts arbitrary ciphertext without the secret key, and is quite efficient: it amounts to applying the LLL algorithm twice on small matrices. Used against *existing practical instantiations* of PIR protocols, it allows the server to recover the users' access pattern in a matter of seconds.

1 Introduction

Cloud computing has gained widespread importance and adoption in recent years. One of the main concerns of cloud security is user privacy. Encryption of data at rest is a first step towards the protection of user data in such a setting. In combination with fully homomorphic encryption [Gen09], cloud servers can continue to provide services to users while only manipulating encrypted data. However, encryption of users' data is only a partial solution to cloud security. Private Information Retrieval (PIR), introduced by Chor, Goldreich, Kushilevitz and Sudan [CKGS98], allows a user to retrieve its data in a manner that prevents the server from knowing which data was retrieved. In a PIR protocol with a single server, the only way to information theoretically hide the users' access pattern is to send the entire data back at each query. (By considering several servers with a copy of the data, secure information theoretic PIR protocols with smaller communication complexity can be achieved.)

In [KO97], Kushilevitz and Ostrovsky presented the first (single database) *computational* PIR (cPIR) where security is achieved against a computationally

bounded server. More generally, they present a construction of a cPIR from an additively homomorphic encryption scheme, i.e. from an encryption scheme that allows to publicly compute an encryption of the sum of the plaintexts from the ciphertexts (see, e.g., [DSH14] for one example). Since this result, numerous protocols of cPIR have been proposed.

In this paper, we focus on the cPIR protocol proposed by Trostle and Parrish at ISC 2010, and more precisely on its underlying (somewhat) additively homomorphic encryption scheme—the TP scheme. Due to its conceptual simplicity and computational efficiency, this scheme was used as a building block of other PIR protocols [BPMÖ12, MBC13, MBC14, EÖM14], and to a private spectrum availability information retrieval protocol [GZL+13]. In particular, it was implemented in Java by Mayberry et al. [MBC13, MBC14] to demonstrate the practicality of PIR in a “real world” setting.

Our Contributions. In this paper we focus on the TP scheme. We present a concrete attack showing that the scheme is not one-way: one can in fact recover the plaintext of any given ciphertext. Our attack is based on the notion of orthogonal lattice introduced by Nguyen and Stern at Crypto ’97 [NS97], and it is very efficient: it amounts to applying LLL reduction twice on lattices of small dimension. We implemented it and carried out the attack on the parameters suggested in [TP10] as well as those used in existing implementations of the TP scheme [MBC13, MBC14]. Every time, it succeeded in a matter of seconds on a desktop computer. Our technique can be described as follows.

The secret key in the TP scheme consists of a pair (b, m) where m is a large secret prime and $b \in \mathbb{Z}_m$ is a secret odd multiplicative mask. A bit μ is encrypted as $c \in \mathbb{Z}_m$ given by

$$c = b \cdot (2r + \mu) \bmod m = b \cdot e + m \cdot k,$$

where r is some random small noise value, $e = 2r + \mu$, and k is quotient in the Euclidean division of c by m .

Now consider a vector $\mathbf{c} \in \mathbb{Z}^t$ of ciphertexts associated with the plaintext vector $\boldsymbol{\mu}$, and let \mathbf{e}, \mathbf{k} be the corresponding vectors of “noisy plaintexts” and Euclidean quotients:

$$\mathbf{c} = b \cdot \mathbf{e} + m \cdot \mathbf{k}.$$

The first step of our attack is similar to [NS98, CNT10]: by applying lattice reduction on the lattice of vectors orthogonal to \mathbf{c} in \mathbb{Z}^t , we can obtain a short basis $\{\mathbf{u}_1, \dots, \mathbf{u}_{t-2}\}$ of the lattice orthogonal to $L = \mathbb{Z}\mathbf{e} \oplus \mathbb{Z}\mathbf{k}$, and taking the orthogonal again, we get a short basis $\{\mathbf{x}, \mathbf{y}\}$ of L . In particular, $\mathbf{e} = u\mathbf{x} + v\mathbf{y}$ for some integers u, v . As a result, $\boldsymbol{\mu} = \mathbf{e} \bmod 2$ is equal modulo 2 to one of $\mathbf{0}$, \mathbf{x} , \mathbf{y} or $\mathbf{x} + \mathbf{y}$, and can thus be recovered with probability at least $1/4$.

We stress that our attack differs from [NS98, CNT10] in at least two respects: on the one hand, the modulus m is secret, which is the main reason why Trostle and Parrish believed their scheme to be secure; and on the other hand, the

vectors \mathbf{x}, \mathbf{y} are actually *too short* to allow us to recover \mathbf{r} and \mathbf{k} (or the integers u, v above) directly, due to an exponentially large search space. To the best of our knowledge, this last point is an unheard of situation in the realm of orthogonal lattice techniques so far, and makes this particular attack quite interesting from a theoretical cryptanalytic viewpoint as well.

Outline. In Sect. 2, we recall Trostle and Parrish’s scheme, some of its applications to PIR and provide some background on orthogonal lattices. In Sect. 3, we describe our attack against the scheme. And finally, we assess its practicality in Sect. 4.

2 Preliminaries

For any integer $n \in \mathbb{Z}$, we denote by $[n]$ the set $\{1, \dots, n\}$. Vectors are denoted in bold characters. For any vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^t$, $\|\mathbf{x}\|$ denotes the Euclidean norm of \mathbf{x} , $[\mathbf{x}]_n = (x_i \bmod n)_{i \in [t]}$ denotes the componentwise reduction of the coefficients of \mathbf{x} modulo n , and $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the scalar product of \mathbf{x} and \mathbf{y} .

2.1 Trostle and Parrish’s SHE Scheme

In this section, we present the secret-key encryption scheme of Trostle and Parrish [TP10], a key ingredient of their PIR protocol.

Let λ be the security parameter, η the bit-length of the secret modulus m and ρ the bit-length of the noise in a fresh ciphertext (both η and ρ are functions of λ).

KeyGen(1^λ). On input the security parameter λ , generate a η -bit secret modulus m , and an odd secret random invertible mask $b \in \mathbb{Z}_m$. Output $\text{sk} = \{m, b\}$.

Encrypt($\text{sk}, \mu \in \{0, 1\}$). On input the secret key $\text{sk} = \{m, b\}$ and a message μ , sample $r \xleftarrow{u} [0, 2^{\rho-1})$ and output $c = b \cdot (2 \cdot r + \mu) \bmod m$.

Decrypt(sk, c). On input the secret key $\text{sk} = \{m, b\}$, a ciphertext c , output $\mu = (b^{-1} \cdot c \bmod m) \bmod 2$.

This scheme is (somewhat) additively homomorphic, i.e. can be used to compute the sum of (a bounded number of) values only manipulating encrypted values. More precisely consider two ciphertexts $c_1 = b \cdot (2r_1 + \mu_1) \bmod m$ and $c_2 = b \cdot (2r_2 + \mu_2) \bmod m$ where r_1 (resp. r_2) is a ρ_1 -bit (resp. ρ_2 -bit) integer. One can homomorphically add the ciphertexts: the ciphertext $c_1 + c_2$ is an encryption of $\mu_1 + \mu_2 \bmod 2$ under a $(\max(\rho_1, \rho_2) + 1)$ -bit noise. Note that the ciphertext noise must remain smaller than m to maintain correctness.

Remark 1. In [TP10], the scheme is also described with message space \mathbb{Z}_N for any $N \geq 2$. An encryption of $\mu \in \mathbb{Z}_N$ is an integer c such that $c = b \cdot (N \cdot r + \mu) \bmod m$ with $r \xleftarrow{u} [0, 2^\rho/N)$, and we recover μ from c by $\mu = (b^{-1} \cdot c \bmod m) \bmod N$. We discuss extension of our attack to this setting in Sect. 3.3.

2.2 Applications to PIR

Due to its simplicity and computational efficiency (the homomorphic addition being a simple addition of integers), the TP scheme is used as building block in several PIR protocols [TP10, GZL+13, MBC13, MBC14, EÖM14]. Below, we briefly describe the original PIR protocol, and the protocols *implemented* by Mayberry et al. [MBC13, MBC14]. In the following, assume a user want to recover a file among t files of bitsize s from a server.

In their initial paper, Trostle and Parrish described the following protocol (we present the variant in which a user wants to recover one row of a database that is a square bit array). The database D is a $t \times s$ matrix of bits, and a user send $\mathbf{c} = (c_1, \dots, c_t)$ to the server, where $c_i \leftarrow \text{Encrypt}(1)$ if the user requests the i -th row of D and $c_j \leftarrow \text{Encrypt}(0)$ for $j \neq i$. The server then multiplies the j -th row by c_j for all j , adds all the rows and sends the result to the user. Since the TP scheme is additively homomorphic, the users recovers a vector \mathbf{c}' such that c'_k encrypts the k -th coefficient of the i -th row of D .

In [MBC13], Mayberry et al. considered the TP scheme with plaintext space \mathbb{Z}_N with $N = 2^\ell$, and use the fact that if $c \leftarrow \text{Encrypt}(1)$ and $\mu \in \mathbb{Z}_N$, then $\mu \cdot c$ encrypts μ (this is a special property of the TP scheme – and could be obtained from any somewhat homomorphic encryption scheme [Gen09] by “encrypting” μ). The database D is a table of $t \times (s/\ell)$ ℓ -bit integers. The rest of the protocol is as above.

Finally, in [MBC14], Mayberry et al. combined the previous approach with an Oblivious RAM protocol to obtain an ORAM-like protocol in which the communication complexity is significantly improved compared to previous ORAM protocols, at the cost of some computational complexity on the server side (coming from the PIR protocol).¹

Note that in all three protocols, a user seeking to recover the i -th row of the database will send a vector of ciphertexts

$$\mathbf{c} = (c_1, \dots, c_t),$$

where c_i encrypts 1 and the c_j 's for $j \neq i$ encrypt 0. Without loss of generality (see Remark 3 page 7), we assume that $N = 2$ and we describe an attack which allows to recover the index of the queried row efficiently.

2.3 The Orthogonal Lattice

In this section, we recall some useful facts about the notion of orthogonal lattice and LLL [NS97, NS01, LLL82].

Let t be an integer. For any vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^t$, we say that \mathbf{u} and \mathbf{v} are orthogonal if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$, and we denote it $\mathbf{u} \perp \mathbf{v}$. For any vector $\mathbf{u} \in \mathbb{Z}^t$, we denote \mathbf{u}^\perp the set of vectors in \mathbb{Z}^t orthogonal to \mathbf{u} . More generally, if L is a

¹ The TP scheme was *one* possible building block of this protocol; therefore the latter might still be secure when instantiated with a different homomorphic encryption scheme.

lattice in \mathbb{Z}^t , its orthogonal lattice L^\perp is defined as the set of vectors in \mathbb{Z}^t orthogonal to the points in L , i.e.

$$L^\perp = \{\mathbf{v} \in \mathbb{Z}^t \mid \forall \mathbf{u} \in L, \langle \mathbf{u}, \mathbf{v} \rangle = 0\}.$$

We have the following theorems [NS97]:

Theorem 1. *If L is a lattice in \mathbb{Z}^t , then $\dim(L) + \dim(L^\perp) = t$.*

Theorem 2. *There exists an algorithm which, given any basis $\{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ of a lattice L in \mathbb{Z}^t of dimension d , outputs an LLL-reduced basis of the orthogonal lattice L^\perp , and whose running time is polynomial with respect to t, d and any upper bound on the bit-length of the $\|\mathbf{b}_j\|$'s.*

Most of the vectors of a reduced basis of L^\perp are quite shorts, with norm around $\det(L^\perp)^{1/(t-\dim(L))}$. In practice, a very simple algorithm for Theorem 2 consists in a single call to LLL [LLL82]; we refer the reader to [NS97] for details, and will use that algorithm in Sect. 4.

3 Breaking the One-Wayness of the Scheme

In this section, we show that the scheme described in Sect. 2.1 is not one-way.

3.1 Overview

Let $\text{sk} = \{m, b\} \leftarrow \text{KeyGen}(1^\lambda)$ be a secret key, and $\mathbf{c} = (c_i)_{i \in [t]} \in \mathbb{Z}^t$ be a vector of ciphertexts such that $c_i \leftarrow \text{Encrypt}(\text{sk}, \mu_i)$ where $\boldsymbol{\mu} = (\mu_i)_{i \in [t]} \in \{0, 1\}^t$. We can write, for each $i \in [t]$:

$$c_i = b \cdot (2r_i + \mu_i) \bmod m = b \cdot e_i + m \cdot k_i$$

with $e_i = 2r_i + \mu_i$ and k_i the quotient in the Euclidean division of $b \cdot e_i$ by m . Thus, if we let $\mathbf{e} = (e_i)_{i \in [t]}$ and $\mathbf{k} = (k_i)_{i \in [t]}$ we have:

$$\mathbf{c} = b \cdot \mathbf{e} + m \cdot \mathbf{k}. \quad (1)$$

Now a rough sketch of the attack is as follows. Consider short vectors $\mathbf{u}_1, \dots, \mathbf{u}_{t-2} \in \mathbb{Z}^t$ orthogonal to \mathbf{c} . For all $j \in [t-2]$, we get that

$$0 = \langle \mathbf{u}_j, \mathbf{c} \rangle = b \cdot \langle \mathbf{u}_j, \mathbf{e} \rangle + m \cdot \langle \mathbf{u}_j, \mathbf{k} \rangle.$$

If the $\|\mathbf{u}_j\|$'s are sufficiently short, the fact that \mathbf{e} and \mathbf{k} are also short yields:

$$\langle \mathbf{u}_j, \mathbf{e} \rangle = 0 \quad \text{and} \quad \langle \mathbf{u}_j, \mathbf{k} \rangle = 0$$

for all j , and hence \mathbf{e} and \mathbf{k} belong to the orthogonal L^\perp of the lattice L spanned by $\mathbf{u}_1, \dots, \mathbf{u}_{t-2}$. Then, if $\{\mathbf{x}, \mathbf{y}\}$ is any basis of L^\perp (easy to find from the \mathbf{u}_j 's), there are only three possible non-zero linear combinations of \mathbf{x} and \mathbf{y} modulo 2 (namely \mathbf{x} , \mathbf{y} and $\mathbf{x} + \mathbf{y}$), and we know that the vector of plaintexts $\boldsymbol{\mu} = \mathbf{e} \bmod 2$ is either one of them or equal to $\mathbf{0}$. The encryption scheme is therefore not one-way.

3.2 Applying Orthogonal Lattice Techniques

The first steps of our attack resemble the attack of Nguyen and Stern [NS98] against the Itoh-Okamoto-Mambo cryptosystem [IOM97], and similar attacks such that the one of Coron et al. on EMV signatures [CNT10]. See also [NT12] for a relevant theoretical discussion. In particular, a simple observation common with those previous attacks is that a vector orthogonal to \mathbf{c} is either large, or orthogonal to both \mathbf{e} and \mathbf{k} .

Lemma 1. *Let $\mathbf{u} \in \mathbb{Z}^t$. If $\mathbf{u} \perp \mathbf{c}$, then ($\mathbf{u} \perp \mathbf{e}$ and $\mathbf{u} \perp \mathbf{k}$), or $\|\mathbf{u}\| \geq m/(t^{1/2} \cdot 2^{\rho+1})$.*

Proof. Let $\mathbf{u} \in \mathbb{Z}^t$ such that $\|\mathbf{u}\| < m/(t^{1/2} \cdot 2^{\rho+1})$ and $\mathbf{u} \perp \mathbf{c}$. We have that $|\langle \mathbf{u}, \mathbf{e} \rangle| \leq \|\mathbf{u}\| \cdot \|\mathbf{e}\| < m$. Now,

$$0 = \langle \mathbf{u}, \mathbf{c} \rangle = b \cdot \langle \mathbf{u}, \mathbf{e} \rangle + m \cdot \langle \mathbf{u}, \mathbf{k} \rangle,$$

and since $\gcd(b, m) = 1$, this yields that $\langle \mathbf{u}, \mathbf{e} \rangle = 0$, and then that $\langle \mathbf{u}, \mathbf{k} \rangle = 0$. \square

From Theorem 2, it is possible to compute a reduced basis $\{\mathbf{u}_1, \dots, \mathbf{u}_{t-1}\}$ of $\mathbf{c}^\perp \subset \mathbb{Z}^t$ of vectors orthogonal to \mathbf{c} in \mathbb{Z}^t . From Lemma 1, we get that for all $j \in [t-1]$, there are two possibilities:

- (1) $\mathbf{u}_j \perp \mathbf{e}$ and $\mathbf{u}_j \perp \mathbf{k}$, in which case \mathbf{u}_j belongs to the lattice $\{\mathbf{e}, \mathbf{k}\}^\perp$ of vectors in \mathbb{Z}^t orthogonal to both \mathbf{e} and \mathbf{k} ;
- (2) $\|\mathbf{u}_j\| \geq m/(t^{1/2} \cdot 2^{\rho+1})$.

Since \mathbf{e} and \mathbf{k} are linearly independent, the first possibility cannot hold for all $j \in [t-1]$ (for reasons of dimensions) and the largest \mathbf{u}_j , say \mathbf{u}_{t-1} , must satisfy $\|\mathbf{u}_{t-1}\| \geq m/(t \cdot 2^{\rho+1})$. Now the other vectors form a lattice $L = \mathbb{Z}\mathbf{u}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{u}_{t-2}$ of rank $t-2$ and of volume

$$V = \text{vol}(L) \approx \frac{\text{vol}(\mathbf{c}^\perp)}{\|\mathbf{u}_{t-1}\|} = \frac{\|\mathbf{c}\|}{\|\mathbf{u}_{t-1}\|} \leq t \cdot 2^{\rho+1},$$

which can heuristically be expected to behave like a random lattice. In particular, assuming the Gaussian heuristic, we should have

$$\|\mathbf{u}_j\| = \mathcal{O}(\sqrt{t-2} \cdot V^{1/(t-2)}) = \mathcal{O}(t^{1/2} \cdot V^{1/(t-2)}) \quad \text{for } j \in [t-2].$$

Thus, the condition for $\mathbf{u}_1, \dots, \mathbf{u}_{t-2}$ all being orthogonal to \mathbf{e}, \mathbf{k} becomes:

$$(t \cdot 2^{\rho+1})^{1+\frac{1}{t-2}} \ll m.$$

Taking logarithms and ignoring logarithmic factors, this means:

$$t \gtrsim 2 + \frac{\rho+1}{\eta - \rho - 1} = \frac{2-\alpha}{1-\alpha} \quad \text{where } \alpha = \frac{\rho+1}{\eta}. \tag{2}$$

Assuming this condition (2) is satisfied, the vectors \mathbf{e} and \mathbf{k} belong to L^\perp . Denote $\{\mathbf{x}, \mathbf{y}\}$ an arbitrary basis of that lattice. Since $\mathbf{e} \in L^\perp$, there exist integers $u, v \in \mathbb{Z}$ such that $\mathbf{e} = u\mathbf{x} + v\mathbf{y}$. This yields

$$\boldsymbol{\mu} = [\mathbf{e}]_2 \in \{\mathbf{0}, [\mathbf{x}]_2, [\mathbf{y}]_2, [\mathbf{x} + \mathbf{y}]_2\},$$

which breaks the one-wayness of the scheme (and in applications to e.g. PIR, the case $\boldsymbol{\mu} = \mathbf{0}$ is excluded, so we really find $\boldsymbol{\mu}$ as one of three possible bit vectors).

Remark 2. It is interesting to note that we can find a (short) basis such that

$$\|\mathbf{x}\|, \|\mathbf{y}\| = \mathcal{O}(\sqrt{2} \cdot V^{1/2}) = \mathcal{O}(t^{1/2} \cdot 2^{\rho/2}).$$

Quite surprisingly these vectors \mathbf{x}, \mathbf{y} (of the “doubly orthogonal” lattice) are actually *too short* to provide a direct break, in the sense that the coefficients (u, v) of \mathbf{e} in the basis $\{\mathbf{x}, \mathbf{y}\}$ of L^\perp are actually exponentially large (of $\approx \rho/2$ bits), so that we cannot hope to recover the vector \mathbf{e} itself from this data.

In fact, \mathbf{e} and \mathbf{k} are in some sense hidden, since for any pair (u', v') of coprime integers of the same size as (u, v) , we can complete the “fake” vector $\mathbf{e}' = u'\mathbf{x} + v'\mathbf{y}$ into a basis $\{\mathbf{e}', \mathbf{k}'\}$ of L^\perp of the correct size, and deduce a “fake” secret key (m', b') also of the correct size such that $\mathbf{c} = b' \cdot \mathbf{e}' + m' \cdot \mathbf{k}'$. This is, to the best of our knowledge, an unheard of situation for orthogonal lattice attacks!

But again, our attack does not need to recover \mathbf{e} completely to break the one-wayness of the scheme. Since the scheme encrypts bits, we only need to recover $[\mathbf{e}]_2$, and that is easy.

3.3 Larger Message Space

As mentioned in Remark 1, instead of \mathbb{Z}_2 , the message space could be \mathbb{Z}_N for $N \geq 2$. Let N_0 be the smallest prime factor of N (if N is prime, $N_0 = N$).

Using the notation of previous section, our attack recovers a basis $\{\mathbf{x}, \mathbf{y}\}$ of L^\perp . Since $\mathbf{e} \in L^\perp$, there exists $u, v \in \mathbb{Z}$ such that $\mathbf{e} = u\mathbf{x} + v\mathbf{y}$. Now there are at most N^2 pairs $(u \bmod N, v \bmod N)$. Therefore, we can recover the plaintext by a random guess with probability at least N^{-2} , and the scheme is therefore not one-way provided that $N = \text{poly}(\lambda)$.

Similarly, if $N_0 = \text{poly}(\lambda)$, the same attack shows that the scheme is not IND-CPA-secure, because for every component μ_i of $\boldsymbol{\mu}$ divisible by N_0 , the corresponding components x_i, y_i of \mathbf{x}, \mathbf{y} are both divisible by N_0 with significant probability $1/N_0^2$, whereas this cannot happen if μ_i is not divisible by N_0 .

Finally, for a superpolynomial choice of N_0 , our attack allows to recover small messages $\boldsymbol{\mu}$. Denote $\mathbf{e} = N \cdot \mathbf{r} + \boldsymbol{\mu}$. If the $\|\mathbf{u}_j\|$'s and $\|\boldsymbol{\mu}\|$ are sufficiently small (e.g. such that $\langle \mathbf{u}_j, \boldsymbol{\mu} \rangle < N$ for all j), then $\mathbf{u}_j \perp \mathbf{e}$ yields $\mathbf{u}_j \perp \mathbf{r}$ and $\mathbf{u}_j \perp \boldsymbol{\mu}$ for all j . Therefore $\boldsymbol{\mu}$ is likely to be the shortest vector of L^\perp and can be efficiently recovered by lattice reduction. Our attack seems only ineffective against a superpolynomial choice of N_0 when encrypting large messages.

Remark 3. In the PIR protocols of [MBC13, MBC14], the message space is chosen to be $N = 2^\ell$ for $\ell \geq 1$. From the discussion above, it follows that one can recover the queried index file to the server.²

4 Implementation of the Attack

Since the attack is heuristic, one needs to assess its behavior in practice. We implemented the attack described in Sect. 3 using SAGE [S+14].

² Note that taking selecting N as a superpolynomial prime does not thwart the attack since the users sends encryption of *bits*.

4.1 Attack Summary

Assume that, for t bits μ_1, \dots, μ_t , we know the ciphertexts c_1, \dots, c_t . Then we can heuristically recover $\boldsymbol{\mu} = (\mu_i)_{i \in [t]}$ as follows.

- (1) Define $\mathbf{c} = (c_1, \dots, c_t) \in \mathbb{Z}^t$.
- (2) Compute an LLL-reduced [LLL82] basis $\{\mathbf{u}_1, \dots, \mathbf{u}_{t-1}\}$ of the lattice $\mathbf{c}^\perp \subset \mathbb{Z}^t$ of vectors in \mathbb{Z}^t orthogonal to \mathbf{c} . This is done by applying LLL to the lattice in \mathbb{Z}^{1+t} generated by the rows of the following matrix:

$$\begin{pmatrix} \gamma \cdot c_1 & 1 & 0 \\ \vdots & \ddots & \\ \gamma \cdot c_t & 0 & 1 \end{pmatrix},$$

where γ is a large constant, and keeping only the t last coefficients of each resulting vector.

- (3) Compute an LLL-reduced basis $\{\mathbf{x}, \mathbf{y}\}$ of the orthogonal L^\perp to the lattice $L = \mathbb{Z}\mathbf{u}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{u}_{t-2} \subset \mathbb{Z}^t$ of rank $t-2$. Again, this amounts at applying LLL to the lattice in \mathbb{Z}^{t-2+t} generated by the rows of

$$\begin{pmatrix} \gamma' \cdot u_{1,1} & \dots & \gamma' \cdot u_{t-2,1} & 1 & 0 \\ \vdots & & \vdots & & \ddots \\ \gamma' \cdot u_{1,t} & \dots & \gamma' \cdot u_{t-2,t} & 0 & 1 \end{pmatrix},$$

where γ' is a large constant, and keeping only the t last coefficients of each resulting vector.

- (4) Output $\mathbf{0}$, $[\mathbf{x}]_2$, $[\mathbf{y}]_2$ and $[\mathbf{x} + \mathbf{y}]_2$.

Heuristically, this attack allows us to guess $\boldsymbol{\mu}$ with probability at least $1/4$. Moreover, if we know $t-1$ coefficients of $\boldsymbol{\mu}$ and have to guess the last one (as in a security game, or in PIR protocols where only one bit is 1), the previous method is likely for large enough t 's to make us guess it with probability 1.

4.2 Experimental Results

We ran our attack against the parameters suggested by Trostle and Parrish [TP10] and the parameters *used* in the proof-of-concept implementations in Java of Mayberry et al. [MBC13, MBC14] – we give these parameters in Table 1.

Table 2a gives the success probability of our attack in function of the parameters and the number of ciphertext t used. As expected when $(\log_2 m - \rho)$ becomes small, one will need more ciphertexts for the attack to be successful. Finally, our attack proves to be really efficient against parameters of Table 1, i.e. parameters used in “real world” implementations of PIR protocols [TP10, MBC13, MBC14] – cf. Table 2b.

Table 1. Parameters sets.

Set of parameters	$\log_2(m)$	ρ
Set-Ia [TP10]	200	188
Set-Ib [TP10]	400	385
Set-IIa [MBC13]	4513	4113
Set-IIb [MBC13]	2195	1155
Set-IIIa [MBC14]	522	384
Set-IIIb [MBC14]	396	296

Table 2. Attack success probability and efficiency for each parameter set, in function of the number t of ciphertexts used for the attack (average value over 500 experiments on a single 3.4Ghz Intel Core i7 CPU).

# ciphertexts t	10	20	40
Set-Ia	0%	0%	100%
Set-Ib	0%	0%	100%
Set-IIa	0%	100%	100%
Set-IIb	100%	100%	100%
Set-IIIa	100%	100%	100%
Set-IIIb	100%	100%	100%

(a) Attack success probability

# ciphertexts t	10	20	40
Set-Ia	—	—	1.45s
Set-Ib	—	—	2.92s
Set-IIa	—	3.51s	38.1s
Set-IIb	88ms	919ms	10.0s
Set-IIIa	28ms	289ms	3.04s
Set-IIIb	23ms	220ms	2.33s

(b) Efficiency of the attack

References

- [BPMÖ12] 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)
- [CKGS98] Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
- [CNT10] Coron, J.-S., Naccache, D., Tibouchi, M.: Fault attacks against EMV signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
- [DSH14] Doröz, Y., Sunar, B., Hammouri, G.: Bandwidth efficient PIR from NTRU. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 195–207. Springer, Heidelberg (2014)
- [EÖM14] Elkhiyaoui, K., Önen, M., Molva, R.: Privacy preserving delegated word search in the cloud. In: Obaidat, M.S., Holzinger, A., Samarati, P. (eds.) SECRYPT 2014, pp. 137–150. SciTePress (2014)
- [Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC 2009, pp. 169–178. ACM (2009)
- [GZL+13] Gao, Z., Zhu, H., Liu, Y., Li, M., Cao, Z.: Location privacy in database-driven cognitive radio networks: attacks and countermeasures. In: INFOCOM 2013, pp. 2751–2759. IEEE (2013)
- [IOM97] Itoh, K., Okamoto, E., Mambo, M.: Proposal of a fast public key cryptosystem. In: Adams, C., Just, M. (eds.) SAC 1997, pp. 224–230 (1997)

- [KO97] Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: FOCS 1997, pp. 364–373. IEEE Computer Society (1997)
- [LLL82] Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261(4), 515–534 (1982)
- [MBC13] Mayberry, T., Blass, E.-O., Chan, A.H.: PIRMAP: efficient private information retrieval for mapreduce. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 371–385. Springer, Heidelberg (2013)
- [MBC14] Mayberry, T., Blass, E.-O., Chan, A.H.: Efficient private file retrieval by combining ORAM and PIR. In: NDSS 2014 (2014)
- [NS97] Nguyen, P.Q., Stern, J.: Merkle-hellman revisited: a cryptanalysis of the Qu-vanstone cryptosystem based on group factorizations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 198–212. Springer, Heidelberg (1997)
- [NS98] Nguyen, P.Q., Stern, J.: Cryptanalysis of a fast public key cryptosystem presented at SAC 1997. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, p. 213. Springer, Heidelberg (1999)
- [NS01] Nguyen, P.Q., Stern, J.: The two faces of lattices in cryptology. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 146–180. Springer, Heidelberg (2001)
- [NT12] Nguyen, P.Q., Tibouchi, M.: Lattice-based fault attacks on signatures. In: Joye, M., Tunstall, M. (eds.) Fault Analysis in Cryptography. Information Security and Cryptography, pp. 201–220. Springer (2012)
- [S+14] Stein, W.A., et al.: Sage Mathematics Software (Version 6.2). The Sage Development Team (2014). <http://www.sagemath.org>
- [TP10] 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)

Homomorphic Computation of Edit Distance

Jung Hee Cheon¹(✉), Miran Kim¹, and Kristin Lauter²

¹ Seoul National University (SNU), Seoul, Republic of Korea

{jhcheon, alfkss500}@snu.ac.kr

² Microsoft Research, Redmond, WA, USA

klauter@microsoft.com

Abstract. These days genomic sequence analysis provides a key way of understanding the biology of an organism. However, since these sequences contain much private information, it can be very dangerous to reveal any part of them. It is desirable to protect this sensitive information when performing sequence analysis in public. As a first step in this direction, we present a method to perform the edit distance algorithm on encrypted data to obtain an encrypted result. In our approach, the genomic data owner provides only the encrypted sequence, and the public commercial cloud can perform the sequence analysis without decryption. The result can be decrypted only by the data owner or designated representative holding the decryption key.

In this paper, we describe how to calculate edit distance on encrypted data with a somewhat homomorphic encryption scheme and analyze its performance. More precisely, given two encrypted sequences of lengths n and m , we show that a somewhat homomorphic scheme of depth $\mathcal{O}((n+m)\log\log(n+m))$ can evaluate the edit distance algorithm in $\mathcal{O}(nm\log(n+m))$ homomorphic computations. In the case of $n = m$, the depth can be brought down to $\mathcal{O}(n)$ using our optimization technique. Finally, we present the estimated performance of the edit distance algorithm and verify it by implementing it for short DNA sequences.

Keywords: Edit distance • Homomorphic encryption • Arithmetic circuit

1 Introduction

In bioinformatics, the term “Sequence Analysis” refers to the process of arranging DNA, RNA, or peptide sequences to understand their structures and features. Relationships between sequences are usually discovered by aligning them appropriately and identifying the most closely matching subsequences. In this paper, we focus on the well-known edit distance algorithm [25], which measures the dissimilarity of two strings. Calculating the edit distance between public reference strings and patients’ DNA sequences can be used to solve the problem of approximate string matching. In practice, there are deployed services to compare DNA sequences. For example, the European Bioinformatics Institute

(EBI) website [6] provides “Bic-SW Database Searches” where one can apply a sequence analysis algorithm to any two DNA sequences (e.g., Smith-Waterman algorithm).

Privacy Threats from Exposing Genomic Data. There are many projects to collect DNA information from participants in order to discover genomic sequences associated with disease susceptibility. The Personal Genome Project displays genotypic and phenotypic information in a public database [21] and the HapMap Project has developed a public repository of genome sequences [12], which means that genomic data has become publicly accessible. However, even anonymized genomic data can leak significant information about the participants (see for example [7,9,23]). In fact, in 2012, an artist created portrait sculptures from analyses of genetic material collected in public places [24]. From some samples, he could infer physical characteristics of strangers such as the gender, eye color, nose size and so on. Secondly, even if DNA sequences are not associated with explicit identifiers such as name, sex, date of birth, or address, one can recover such personal data using re-identification methods: genotype-phenotype inference [19], location-visit patterns [20], family structure [10], and dictionary attacks. Thus, DNA sequences are sensitive and valuable enough that we should not reveal our own sequences even when performing sequence analysis.

Privacy through Encryption. In this work, we consider the potential for using homomorphic encryption to protect privacy in genomic computations. Compared with MPC protocols based on recent optimizations of garbled circuit techniques [11,14], homomorphic encryption is often considered to be slower and less efficient. But homomorphic encryption has a number of other advantages, allowing for more flexible scenarios and functionality and requiring less interaction, thereby reducing communication complexity. Typically no interaction is required for applications of (single-key) homomorphic encryption. Also, homomorphic encryption schemes have become more practical recently, due to a number of improvements, including techniques which avoid the costly bootstrapping procedure for fixed computations, such as using leveled or somewhat homomorphic encryption (SWHE) schemes.

Scenarios. Homomorphic encryption allows the data owner to upload encrypted data to a cloud service. The cloud service can operate on the encrypted data without requiring the decryption key or any interaction with the data owner. The service returns the encrypted results to the data owner, who can decrypt using the secret key. A cloud provider could thus provide Direct-to-patient services in encrypted form, such as the service mentioned above provided by EBI (Fig. 1).

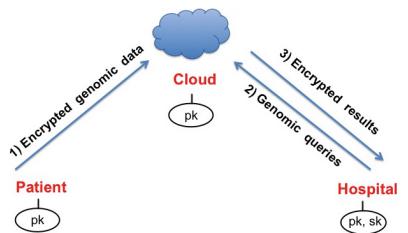


Fig. 1. Scenario of proposed system

As an extension to the scenario, additional functionality can be achieved using public key homomorphic encryption schemes by allowing third parties to upload data directly to the cloud service, encrypted using the public key of the data owner. This scenario could be of interest in situations relevant to genomic computation: for example the data owner is a hospital or clinic, and the third parties are patients or other healthcare providers for those patients. The hospital would like to use the cloud service for analyzing lots of patients. Auxiliary data (from tests, genome sequencing, etc.) can be uploaded to the service using the public key of the hospital. Computations on the encrypted data, such as comparing DNA sequences, output encrypted results which can be decrypted by the hospital or clinic. The secrecy of DNA sequences in the cloud can be protected under the semantic security of homomorphic encryption scheme.

Our Contributions. In this paper, we first describe the homomorphic evaluation of the edit distance algorithm which was suggested by Wagner and Fischer [25]. We show that the algorithm can be implemented on two encrypted sequences of lengths n and m with a somewhat homomorphic scheme of depth $\mathcal{O}((n + m) \log(\log(n + m)))$ in $\mathcal{O}(nm \log(n + m))$ homomorphic computations. Moreover, we introduce an optimization technique to reduce the depth required to implement the algorithm: Divide the edit distance matrix into sub-blocks of size- $(\tau + 1)$ and solve the edit distance problem in each block. We can compute each of them diagonally, consuming $\mathcal{O}(\tau)$ levels in one diagonal-round. Namely, evaluating the circuits in each cell can be processed by a somewhat homomorphic encryption of a constant depth. In particular, in the case of $n = m$, it suffices to compute only a little part of the sub-blocks, so the depth can be brought down to $\mathcal{O}(n)$.

Finally, we estimate the running time of the proposed algorithm for a large n and verify it by implementing it for short DNA sequences. For two encrypted DNA sequences of length 50, we expect that the algorithm would run in one day when estimated based on the recent CCK+ scheme [4]. We also demonstrate the experimental result that it takes about 27.5 sec. for $n = m = 8$ using the GHS scheme [8].

Related Works. Since Wagner and Fischer [25] introduced the problem of determining the edit distance between two strings and presented an algorithm for calculating the distance, there have been a number of approaches for private computation of the distance. In 2003, Atallah et al. [1] proposed a privacy-preserving protocol using an additive homomorphic encryption scheme and oblivious transfers, which had expensive computational and communication costs. Given two strings of lengths n and m , the number of iterations is equal to nm and the total online computational cost is $\mathcal{O}(nm \log(n + m))$. In 2008, Jha et al. [14] presented a more practical privacy-preserving protocol to compute the edit distance with Yao’s “garbled circuits” method [18, 26], and it was improved by Huang et al. [11]. Their computation cost is tractable, but their protocol requires a lot of interactions (e.g., $\mathcal{O}(nm \log(n + m))$ oblivious transfers for Protocol 2 in [14]).

On the other hand, there is prior art on analyzing genomic data using homomorphic encryption. Some of the work is based on additively homomorphic encryption schemes: Kantarcioğlu et al. [15], Kolesnikov et al. [16], and Ayday et al. [2]. In [15], they presented a novel cryptographic framework that allows organizations to support data mining without violating the privacy of the genomic sequences, and in particular they used the Paillier cryptosystem for experimental analysis. The garbled circuit protocols of [16] were given for secure computation of the minimum distance (Hamming distance and Euclidean distance). In [2], they proposed a “privacy-preserving disease susceptibility test” on encrypted genomic data using a modified Paillier cryptosystem. Meanwhile, Cristofaro et al. [5] presented an efficient and secure protocol called “Size- and position-hiding private substring matching” based on a multiplicative homomorphic ElGamal variant so as to check for the presence of DNA markers. Finally, Yasuda et al. [27] gave a practical solution for computation of multiple Hamming distance values using the LNV scheme [17], so that they could find the locations where a pattern occurs in a text. By contrast, the aim of this paper is to compute edit distance on encrypted sequences under somewhat homomorphic encryption schemes (which support additions and a limited number of multiplications of encrypted inputs). Besides DNA sequence analysis, edit distance has many other applications such as spelling correction or determining the longest common subsequences of two strings.

Outline. In Sect. 2, we review the main concept of homomorphic encryption and explain the edit distance algorithm. Section 3 presents the basic circuit building blocks for equality, comparison, and addition. Next, in Sect. 4, we describe our encrypted edit distance algorithm using these primitive circuits and give the analysis of our method. We also introduce optimizations to reduce the depth of implementing the algorithm. Finally, in Sect. 5, we estimate the performance of the proposed algorithm for large DNA sequences and present the real performance for our implementation of the algorithm for short sequences.

2 Preliminaries

In this section, we briefly review the concept of homomorphic encryption and describe the edit distance algorithm which is a measure to quantify the dissimilarity of two strings.

2.1 Homomorphic Encryption

We will encrypt bit-by-bit in this paper, so consider the concept of homomorphic encryption in this respect. For $x \in \{0, 1\}$, we denote the encryption of x by \bar{x} or $\text{Enc}(x)$. Let \oplus and \wedge be the XOR and AND gate, each of which corresponds to addition and multiplication over \mathbb{Z}_2 , respectively. Also, we let $+$ and \times

denote homomorphic addition and multiplication over encrypted data. Then a homomorphic encryption $\text{Enc}(\cdot)$ satisfies the following properties:

$$\text{Enc}(x \oplus y) = \text{Enc}(x) + \text{Enc}(y), \quad \text{Enc}(x \wedge y) = \text{Enc}(x) \times \text{Enc}(y).$$

In our paper, we focus on SWHE schemes for which additions are essentially free and a limited number of multiplications are supported. In particular, SWHE schemes [3,8] use a practical noise-management technique-*modulus switching*, which scales down the ciphertext after every multiplication to reduce the noise by its scaling factor. When we say the (multiplicative) *depth* $\mathbf{D}(\mathbf{C})$ of a circuit \mathbf{C} under homomorphic encryption, it means the total number of reduced levels in the circuit that is being evaluated homomorphically.

2.2 Edit Distance

Assume that there are two strings $\alpha = \alpha_1 \dots \alpha_n$ and $\beta = \beta_1 \dots \beta_m$ over an alphabet Σ . One can make another string with the same length by inserting spaces “-”, called *gaps*, and consider a matrix having two rows with these new strings. A gap in the first (resp. second) row is called **Insertion** (resp. **Deletion**). A column with the same (resp. distinct) characters is called **Match** (resp. **Mismatch**). Then the edit distance between two strings is the minimum number of these edit operations needed to transform one string into the other. More specifically, for two characters α_i and β_j , let us define $t_{i,j}$ as follows:

$$t_{i,j} = \begin{cases} 0 & \text{if } \alpha_i = \beta_j \text{ (Match),} \\ 1 & \text{if } \alpha_i \neq \beta_j \text{ (Mismatch).} \end{cases}$$

In Algorithm 1, we describe the *Wagner-Fischer edit distance algorithm* [25], and the edit distance is simply $D_{n,m}$.

Algorithm Edit distance

Input: $\alpha = \alpha_1 \dots \alpha_n$ and $\beta = \beta_1 \dots \beta_m$

- 1: **for** $i \leftarrow 0$ to n **do**
- 2: $D_{i,0} \leftarrow i$;
- 3: **end for**
- 4: **for** $j \leftarrow 0$ to m **do**
- 5: $D_{0,j} \leftarrow j$;
- 6: **end for**
- 7: **for** $i \leftarrow 1$ to n **do**
- 8: **for** $j \leftarrow 1$ to m **do**
- 9: $t \leftarrow (\alpha_i = \beta_j)? 0 : 1$;
- 10: $D_{i,j} \leftarrow \min\{D_{i-1,j-1} + t, D_{i,j-1} + 1, D_{i-1,j} + 1\}$;
- 11: **end for**
- 12: **end for**
- 13: **return** $D_{n,m}$

3 Circuit Building Blocks

In this section, we present the basic circuit building blocks for computing the edit distance: equality circuit (for checking the equality of two numbers so as to determine match/mismatch of two characters), comparison circuit, and addition circuits. Since it may assume that we can evaluate homomorphic additions for free, it suffices to count the number of multiplication gates sequentially in order to compute the depth of a homomorphic encryption scheme. Thus, we focus on minimizing the number of sequential multiplication gates for circuits so that we can implement them efficiently.

For a circuit C , we denote the number of homomorphic additions and multiplications by $\mathbf{HA}(C)$ and $\mathbf{HM}(C)$. Note that addition with a constant is faster than a classical homomorphic addition, so those are not counted in the number of the homomorphic additions. In Tables 1, 2, and 4, the depth of homomorphic encryption is cumulative while the number of homomorphic computations is not cumulative.

We will express an unsigned μ -bit integer in its binary representation $x_\mu \dots x_1$ and denote the i -th coordinate of x by x_i (or $x[i]$). Then the encryption of x means $\{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_\mu\}$.

3.1 Equality Circuit

A binary circuit for checking the equality of two μ -bit values is defined to have value 1 if the inputs are the same and 0 otherwise. Then it can be written as an arithmetic circuit $\text{EQU}(x, y) = \wedge_{i=1}^\mu (1 \oplus x_i \oplus y_i)$. Using a binary tree, we give the required depth and complexity in Table 3 where \log is the binary logarithm.

3.2 Comparison Circuit

For two unsigned μ -bit values x and y , the comparison circuit is defined by

$$\text{COM}(x, y) = \begin{cases} 0 & \text{if } x \geq y, \\ 1 & \text{otherwise,} \end{cases}$$

and this is written recursively as $\text{COM}(x, y) := c_\mu$ where $c_i = ((x_i \oplus 1) \wedge y_i) \oplus ((x_i \oplus 1 \oplus y_i) \wedge c_{i-1})$ for $i \geq 2$ with an initial value $c_1 = (x_1 \oplus 1) \wedge y_1$. In Table 1, we provide a pseudocode description of this circuit together with an approximation of the levels that it consumes during these operations. Unlike the other steps, the fourth cannot be computed simultaneously for each i , so it consumes linear levels and we have $\mathbf{D}(\text{COM}) = \mu$. On the other hand, the comparison circuit can be evaluated homomorphically with a logarithmic depth, which is formally captured in Lemma 1 below.

Lemma 1. *The Comparison circuit of Table 2 can be evaluated homomorphically on two μ -bits with a somewhat homomorphic encryption of depth $\log(\mu - 1) + 1$ in $\mathcal{O}(\mu \log \mu)$ homomorphic computations.*

Table 1. Pseudocode of COM between two μ -bit values and its complexity

Comparison circuit	Depth of hom. enc.	HA	HM
Input: fresh ciphertexts \bar{x}_i, \bar{y}_j	0	—	—
1. compute $\bar{x}_i + 1$ for $i = 1, \dots, \mu$	0	—	—
2. $\bar{x}_{i1} \leftarrow (\bar{x}_i + 1) + \bar{y}_i$ for $i = 2, \dots, \mu$	0	$\mu - 1$	—
3. $\bar{x}_{i2} \leftarrow (\bar{x}_i + 1) \times \bar{y}_i$ for $i = 1, \dots, \mu$ (in particular, let $\bar{c}_1 \leftarrow \bar{x}_{12}$)	1	—	μ
4. $\bar{c}_i \leftarrow \bar{x}_{i1} + \bar{x}_{i2} \times \bar{c}_{i-1}$ for $i = 2, \dots, \mu$	μ	$\mu - 1$	$\mu - 1$
Total	μ	$2\mu - 2$	$2\mu - 1$

Proof. We consider the comparison circuit as the following expression:

$$\text{COM}(x, y) = d_1 \oplus d_2 \oplus \dots \oplus d_\mu$$

where $d_i = (x_i \oplus 1) \wedge y_i \wedge (\wedge_{j=i+1}^{\mu} (x_j \oplus 1 \oplus y_j))$. From now, the following arguments are underlying ciphertexts for the above circuit. For simplicity, we denote $z_i := (\bar{x}_i + 1) + \bar{y}_i$ for $i = 2, \dots, \mu$, and HM_i the number of homomorphic multiplications to evaluate $\prod_{j=i+1}^{\mu} z_j$ for $i = 1, \dots, \mu - 2$.

We first construct a binary tree of product with $\{z_2, \dots, z_\mu\}$. Then the total number of multiplications to proceed recursively with each of the two nodes is

$$1 + 2 + 4 + \dots + \frac{\mu - 1}{2} \approx \mu - 2,$$

and it needs $\log(\mu - 1)$ levels. We observe that $\prod_{j=i+1}^{\mu} z_j$ has been computed if the number to be multiplied by is in the form of powers of 2 or $\mu - 1$.

Now, we consider the case of $i \in \{1, 2, \dots, \mu - 2\}$ with $\mu - i \neq 2^1, 2^2, \dots, 2^{\lfloor \log(\mu-1) \rfloor}, \mu - 1$. It is true that $\mu - i$ is uniquely written as $2^{k_{i1}} + 2^{k_{i2}} + \dots + 2^{k_{il}}$ where k_{ij} 's are increasing nonnegative numbers. Denote $\mu_{ir} := \mu - (2^{k_{il}} + 2^{k_{il-1}} + \dots + 2^{k_{ir+1}} + 2^{k_{ir}})$ for $1 \leq r \leq l$ and $\mu_{i,l+1} = \mu$, then we have

$$\prod_{j=i+1}^{\mu} z_j = \prod_{r=1}^l (z_{\mu_{ir}+1} z_{\mu_{ir}+2} \cdots z_{\mu_{ir+1}}).$$

Since all $z_{\mu_{ir}+1} z_{\mu_{ir}+2} \cdots z_{\mu_{ir+1}}$'s have been computed as above, what we have to do is just to multiply them each other, which requires $\log l$ levels and $(l - 1)$ homomorphic multiplications. From these observations, we see that

$$\sum_{2^{t-1} < u - i < 2^t} \text{HM}_i = \sum_{l=1}^{t-1} l \cdot \binom{t-1}{l} = (t-1) \cdot 2^{t-2}$$

for $t \in \{2, 3, \dots, \lfloor \log(\mu - 1) \rfloor\}$. So we have

$$\begin{aligned} \sum_{i=1}^{\mu-2} \mathbf{HM}_i &= \sum_{u-i=2^1, 2^2, \dots, \mu-1} \mathbf{HM}_i + \sum_{t=2, 3, \dots, \lfloor \log(\mu-1) \rfloor} \left(\sum_{2^{t-1} < u-i < 2^t} \mathbf{HM}_i \right) \\ &\approx (\mu - 2) + \sum_{t=2, 3, \dots, \lfloor \log(\mu-1) \rfloor} (t - 1) \cdot 2^{t-2} \\ &= \frac{(\mu - 1) \log(\mu - 1)}{2} - 2. \end{aligned}$$

Therefore, as described in Table 2, evaluating the COM circuit can be accomplished using

$$\mu + \left(\frac{(\mu - 1) \log(\mu - 1)}{2} - 2 \right) + (\mu - 1) = 2\mu - 3 + \frac{(\mu - 1) \log(\mu - 1)}{2}$$

homomorphic multiplications with a SWHE scheme of depth $\log(\mu - 1) + 1$. \square

In the following, we show that the comparison circuit leads to the minimal circuits.

Table 2. Pseudocode of COM between two μ -bit values and its complexity

Comparison circuit	Depth of hom. enc.	HA	HM
Input: fresh ciphertexts \bar{x}_i, \bar{y}_j	0		
1. compute $\bar{x}_i + 1$ for $1 \leq i \leq \mu$	0	—	—
2. $\bar{d}_i \leftarrow (\bar{x}_i + 1) \times \bar{y}_i$ for $1 \leq i \leq \mu$	1	—	μ
3. $z_i \leftarrow (\bar{x}_i + 1) + \bar{y}_i$ for $2 \leq i \leq \mu$	0	$\mu - 1$	—
4. $\prod_{j=i+1}^{\mu} z_j$ for $1 \leq i \leq \mu - 2$	$\log(\mu - 1)$	—	$\frac{(\mu-1) \log(\mu-1)}{2} - 2$
5. $\bar{d}_i \leftarrow \bar{d}_i \times \prod_{j=i+1}^{\mu} z_j$ for $1 \leq i \leq \mu - 1$	$\log(\mu - 1) + 1$	—	$\mu - 1$
6. $\overline{\text{COM}}(x, y) \leftarrow \bar{d}_1 + \dots + \bar{d}_{\mu}$	—	$\mu - 1$	—
Total	$\log(\mu - 1) + 1$	$2\mu - 2$	$2\mu - 3 + \frac{(\mu-1) \log(\mu-1)}{2}$

Lemma 2. Given two μ -bit values $x = x_{\mu} \dots x_1$ and $y = y_{\mu} \dots y_1$, then $z = z_{\mu} \dots z_1$ is the minimum value of x and y where

$$z_i = (\text{COM}(x, y) \wedge x_i) \oplus (1 \oplus \text{COM}(x, y) \wedge y_i).$$

Proof. Let us denote a multiplication over integers by “.”. Then it is true that

$$\begin{aligned}\min\{x, y\} &= \text{COM}(x, y) \cdot x + (1 \oplus \text{COM}(x, y)) \cdot y \\ &= \text{COM}(x, y) \cdot \left(\sum_{i=1}^{\mu} x_i \cdot 2^{i-1} \right) + (1 \oplus \text{COM}(x, y)) \cdot \left(\sum_{i=1}^{\mu} y_i \cdot 2^{i-1} \right) \\ &= \sum_{i=1}^{\mu} ((\text{COM}(x, y) \cdot x_i) + ((1 \oplus \text{COM}(x, y)) \cdot y_i)) \cdot 2^{i-1},\end{aligned}$$

where the inputs x and y can be written as binary representations in the second line. Since “ $\text{COM}(x, y) \cdot x_i$ ” and “ $(1 \oplus \text{COM}(x, y)) \cdot y_i$ ” cannot simultaneously be “1”, the lemma follows. \square

From Lemma 2, we define minimum circuits $\text{MIN}^2 = (\text{MIN}_1^2, \dots, \text{MIN}_{\mu}^2)$ by

$$\text{MIN}_i^2 = (\text{COM}(x, y) \wedge x_i) \oplus ((1 \oplus \text{COM}(x, y)) \wedge y_i).$$

Then one can evaluate these circuits homomorphically with a SWHE scheme of depth $(\log(\mu - 1) + 2)$. We also obtain a natural generalization of computing the minimum value between many numbers: apply repeatedly the minimum circuits. Then this naive method has $\mathbf{D}(\text{MIN}^2) = (\log(\mu - 1) + 2) \cdot (\log k)$.

On the other hand, we consider another way to compute the minimum value which requires circuits of lesser depth: Given μ -bit values x^1, \dots, x^k , we define $\text{MIN}^k = (\text{MIN}_1^k, \dots, \text{MIN}_{\mu}^k)$ by $\text{MIN}_i^k = \bigoplus_{j=1}^k (\mathfrak{c}_j \wedge x_i^j)$ where

$$\mathfrak{c}_j = \begin{cases} \text{COM}(x^1, x^2) \wedge \dots \wedge \text{COM}(x^1, x^k) & \text{if } j = 1, \\ (1 \oplus \text{COM}(x^1, x^j)) \wedge \dots \wedge (1 \oplus \text{COM}(x^{j-1}, x^j)) \wedge \text{COM}(x^j, x^{j+1}) \wedge \dots \wedge \text{COM}(x^j, x^k) & \text{if } 2 \leq j < k, \\ (1 \oplus \text{COM}(x^1, x^k)) \wedge \dots \wedge (1 \oplus \text{COM}(x^{k-1}, x^k)) & \text{if } j = k. \end{cases}$$

It is easy to show that this method has

$$\begin{aligned}\mathbf{D}(\text{MIN}^k) &= \log(\mu - 1) + \log(k - 1) + 2, \\ \mathbf{HM}(\text{MIN}^k) &= \left(2\mu - 3 + \frac{(\mu - 1) \log(\mu - 1)}{2} \right) \frac{(k - 1)(k - 2)}{2} + k(k - 2 + \mu).\end{aligned}$$

3.3 Addition Circuits

For two unsigned μ -bit values x and y , we assume that their sum over the integers is less than 2^{μ} , say $s_1 + \dots + s_{\mu} \cdot 2^{\mu-1}$. Then the standard method to add them is the *Ripple-carry* adder such that $\text{ADD}(x, y)$ is defined by (s_1, \dots, s_{μ}) satisfying

$$s_i = \begin{cases} x_1 \oplus y_1 & \text{if } i = 1, \\ x_i \oplus y_i \oplus e_{i-1} & \text{otherwise,} \end{cases} \quad e_i = \begin{cases} x_1 \wedge y_1 & \text{if } i = 1, \\ (x_i \wedge y_i) \oplus ((x_i \oplus y_i) \wedge e_{i-1}) & \text{otherwise.} \end{cases}$$

From now, the k -th value s_k of the sum is denoted by $\text{ADD}(x, y)[k]$. Table 3 reports the required depth and its complexity analysis.

Table 3. Complexity of primitive circuits between two μ -bit values

Circuit	Depth of hom. enc.	HA	HM
EQU	$\log \mu$	μ	$\mu - 1$
COM	$\log(\mu - 1) + 1$	$2\mu - 2$	$2\mu - 3 + \frac{(\mu-1)\log(\mu-1)}{2}$
ADD	$\mu - 1$	$3\mu - 3$	$2\mu - 3$

4 Encrypted Edit Distance Algorithm

We now describe how to execute the homomorphic computation of the edit distance algorithm with regards to the primitive circuits and analyze the performance of our encrypted edit distance algorithm.

Let $|\Sigma|$ be the size of a alphabet and denote $\omega = \lceil \log |\Sigma| \rceil$. As mentioned before, let α and β be two strings over ω -bit alphabet. Then each character of the strings can be seen as an ω -bit value. Suppose that each of them is given encrypted bit-by-bit through a homomorphic encryption.

4.1 Encrypted Edit Distance Algorithm

Since all the values $D_{i,j}$'s are less than $n + m - 1$, we may assume that they are $\lceil \log(n + m - 1) \rceil$ -bits, say μ . Suppose that we have computed $D_{i-1,j-1}$, $D_{i,j-1}$, $D_{i-1,j}$, and ω -bit characters α_i and β_j . From the fact that $t_{i,j} = \text{EQU}(\alpha_i, \beta_j) \oplus 1$, we know

$$\begin{aligned} (D_{i-1,j-1} + t_{i,j})[k] &= ((t_{i,j} \oplus 1) \wedge D_{i-1,j-1}[k]) \oplus (t_{i,j} \wedge \text{ADD}(D_{i-1,j-1}, 1)[k]) \\ &= (\text{EQU}(\alpha_i, \beta_j) \wedge D_{i-1,j-1}[k]) \oplus ((\text{EQU}(\alpha_i, \beta_j) \oplus 1) \wedge \text{ADD}(D_{i-1,j-1}, 1)[k]) \end{aligned}$$

for $1 \leq k \leq \mu$ and

$$\text{ADD}(D_{i-1,j-1}, 1)[k] = \begin{cases} D_{i-1,j-1}[1] \oplus 1 & \text{if } k = 1, \\ D_{i-1,j-1}[k] \oplus (\bigwedge_{l=1}^{k-1} D_{i-1,j-1}[l]) & \text{if } 2 \leq k \leq \mu. \end{cases}$$

In the same way as in Sect. 3.2, $\text{ADD}(D_{i-1,j-1}, 1)$ can be implemented with a SWHE scheme of depth $\log(\mu - 1)$ in μ additions and $\left(\frac{(\mu-1)\log(\mu-1)}{2} - 2\right)$ multiplications since we only need to compute $\prod_{l=1}^{k-1} \text{Enc}(D_{i-1,j-1}[l])$. From these observations, $D_{i,j} = \min\{D_{i-1,j-1} + t_{i,j}, D_{i,j-1} + 1, D_{i-1,j} + 1\}$ can be written as arithmetic circuits using the above circuits. Hence, given ciphertexts $\text{Enc}(D_{i-1,j-1})$, $\text{Enc}(D_{i,j-1})$, $\text{Enc}(D_{i-1,j})$, $\text{Enc}(\alpha_i)$, and $\text{Enc}(\beta_j)$, one can apply these operations so as to compute the encryption of $D_{i,j}$. Continuing this way, we obtain the encrypted edit distance $\text{Enc}(D_{n,m})$.

Table 4. Pseudocode of computing the encrypted value $D_{i,j}$ and its complexity ($\mu = \log(n + m - 1)$)

Binary Circuit	Depth of Hom. Enc.	HA	HM
Input: $D_{i-1,j-1}, D_{i,j-1}, D_{i-1,j}, \alpha_i, \beta_j$			
1. $t \leftarrow \text{EQU}(\alpha_i, \beta_j)$ and compute $t \oplus 1$	$\mathbf{D}(\text{EQU})$	$\mathbf{HA}(\text{EQU})$	$\mathbf{HM}(\text{EQU})$
2. compute $\text{ADD}(D_{i-1,j-1}, 1), \text{ADD}(D_{i,j-1}, 1), \text{ADD}(D_{i-1,j}, 1)$	$\mathbf{D}(\text{ADD})$	$3\mathbf{HA}(\text{ADD})$	$3\mathbf{HM}(\text{ADD})$
3. for $k = 1, \dots, \mu$,			
$D_{i-1,j-1}[k] \leftarrow (t \wedge D_{i-1,j-1}[k]) \oplus ((t \oplus 1) \wedge \text{ADD}(D_{i-1,j-1}, 1)[k])$	$1 + \mathbf{D}(\text{ADD})$	μ	2μ
$D_{i,j-1}[k] \leftarrow \text{ADD}(D_{i,j-1}, 1)[k]$	$\mathbf{D}(\text{ADD})$	—	—
$D_{i-1,j}[k] \leftarrow \text{ADD}(D_{i-1,j}, 1)[k]$	$\mathbf{D}(\text{ADD})$	—	—
4. $c_1 \leftarrow \text{COM}(D_{i-1,j-1}, D_{i,j-1})$	$1 + \mathbf{D}(\text{ADD}) + \mathbf{D}(\text{COM})$	$\mathbf{HA}(\text{COM})$	$\mathbf{HM}(\text{COM})$
$c_2 \leftarrow \text{COM}(D_{i-1,j-1}, D_{i-1,j})$	$1 + \mathbf{D}(\text{ADD}) + \mathbf{D}(\text{COM})$	$\mathbf{HA}(\text{COM})$	$\mathbf{HM}(\text{COM})$
$c_3 \leftarrow \text{COM}(D_{i,j-1}, D_{i-1,j})$	$\mathbf{D}(\text{ADD}) + \mathbf{D}(\text{COM})$	$\mathbf{HA}(\text{COM})$	$\mathbf{HM}(\text{COM})$
5. $c_1 \leftarrow c_1 \wedge c_2, c_2 \leftarrow (1 \oplus c_1) \wedge c_3, c_3 \leftarrow (1 \oplus c_2) \wedge (1 \oplus c_3)$	$2 + \mathbf{D}(\text{ADD}) + \mathbf{D}(\text{COM})$	—	3
6. for $k = 1, \dots, \mu$,			
$D_{i,j}[k] \leftarrow (c_1 \wedge D_{i-1,j-1}[k]) \oplus (c_2 \wedge D_{i,j-1}[k]) \oplus (c_3 \wedge D_{i-1,j}[k])$	$3 + \mathbf{D}(\text{ADD}) + \mathbf{D}(\text{COM})$	2μ	3μ
Total	$3 + \mathbf{D}(\text{ADD}) + \mathbf{D}(\text{COM})$	$\mathbf{HA}(\text{EQU}) + 3\mathbf{HA}(\text{ADD}) + 3\mathbf{HA}(\text{COM}) + 3\mu$	$\mathbf{HM}(\text{EQU}) + 3\mathbf{HM}(\text{ADD}) + 3\mathbf{HM}(\text{COM}) + 5\mu + 3$

4.2 Performance Analysis of Encrypted Edit Distance Algorithm

In Table 4, we describe a pseudocode for obtaining the encrypted value $D_{i,j}$, and provide an approximation of the levels and computational complexity during homomorphic operations. By the building block algorithms of **COM** (in Lemma 1) and **ADD** (in Sect. 4.1), the one diagonal-round circuits have

$$\mathbf{D} = 2\log(\mu-1)+4, \quad \mathbf{HA} = 15\mu+\omega-6, \quad \mathbf{HM} = 3(\mu-1)\log(\mu-1)+11\mu+\omega-13.$$

It is possible to compute $D_{i,j}$'s simultaneously when $i + j$ is a fixed value from $1, 2, \dots, (n + m - 1)$, so we expect to consume $(2\log(\mu-1)+4) \cdot (n + m - 1)$ levels for computing them diagonally, which requires $(15\mu + \omega - 6)nm$ homomorphic additions and $(3(\mu-1)\log(\mu-1)+11\mu+\omega-13)nm$ multiplications in total. In other words, given two encrypted sequences of lengths n and m , a SWHE scheme of depth $\mathcal{O}((n + m)\log(\log(n + m)))$ can evaluate the edit distance algorithm in $\mathcal{O}(nm\log(n + m))$ homomorphic computations.

Remark 1. Lemma 1 shows that we can compare two μ -bits with a circuit of depth $\log \mu$ using a homomorphic bit-encryption scheme. If we consider a large integer ring \mathbb{Z}_t as a message space instead of a binary field, an addition is performed with a degree-1 circuit. However, one can compute the equality circuit via the following method: $\text{EQU}(x, y) = 1 - (x - y)^{t-1}$ for a prime t . Then this circuit has $\mathbf{D}(\text{EQU}) \approx \log t \approx \log(n + m)$ using the square-and-multiply algorithm. Moreover, the comparison algorithm seems to require a circuit of at least depth $\log t$. This implies that a large message space increases the depth of one diagonal-round circuits to $\mathcal{O}(\log(n + m))$, so the edit distance algorithm can be evaluated with a SWHE scheme of depth $\mathcal{O}((n + m)\log(n + m))$.

4.3 Optimization of Encrypted Edit Distance Algorithm

We present an optimization to reduce the depth during the homomorphic evaluations of the algorithm. Let us consider the 3×3 block B in Fig. 2.

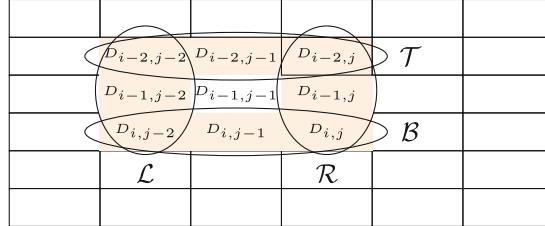


Fig. 2. Block of size 3

It is true that if we have computed the top and left values of this block, $D_{i-2,j-2}$, $D_{i-2,j-1}$, $D_{i-2,j}$, $D_{i-1,j-2}$, $D_{i,j-2}$, then all other values can be expressed in terms of them. For example, $D_{i,j}$ is the minimum value between the following 7 numbers:

$$\begin{aligned} & D_{i-2,j-2} + t_{i-1,j-1} + t_{i,j}, \quad D_{i-2,j-1} + t_{i-1,j} + 1, \quad D_{i-2,j-1} + t_{i-1,j} + 1, \\ & D_{i-1,j-2} + t_{i,j-1} + 1, \quad D_{i-1,j-2} + t_{i,j-1} + 1, \quad D_{i-2,j} + 2, \quad D_{i,j-2} + 2. \end{aligned}$$

In general, we consider a block of size- $(\tau + 1)$ which consists of the following sets:

$$\begin{aligned} \text{top : } \mathcal{T} &= \{D_{i-\tau,j-\tau}, D_{i-\tau,j-\tau+1}, \dots, D_{i-\tau,j}\}, \\ \text{left : } \mathcal{L} &= \{D_{i-\tau,j-\tau}, D_{i-\tau+1,j-\tau}, \dots, D_{i,j-\tau}\}, \\ \text{right : } \mathcal{R} &= \{D_{i-\tau,j}, D_{i-\tau+1,j}, \dots, D_{i,j}\}, \\ \text{bottom : } \mathcal{B} &= \{D_{i,j-\tau}, D_{i,j-\tau+1}, \dots, D_{i,j}\}. \end{aligned}$$

Then all the values of \mathcal{R} and \mathcal{B} are expressed in terms of values of \mathcal{T} and \mathcal{L} .

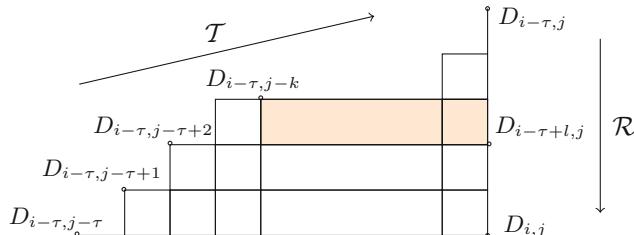


Fig. 3. Grid of size- $(\tau + 1)$ block

More precisely, consider the grid shown in Fig. 3. One can only move one unit right or down on the grid: if moving right from $D_{i-k,j-l}$, then $t_{i-k+1,j-l+1}$ is added to the value and we obtain $D_{i-k,j-l} + t_{i-k+1,j-l+1}$. In the case of moving one unit down, “1” is added to it. We note that the number of shortest paths from $D_{i-\tau,j-k}$ to $D_{i-\tau+l,j}$ is $\frac{l!}{k!(l-k)!} = \binom{l}{k}$ for some $l \geq k$ since the paths include k steps in the x axis and $(l-k)$ steps in y axis. It is seen as the number of the functions of $D_{i-\tau+l,j}$ in terms of $D_{i-\tau,j-k}$. From these observations, $D_{i-\tau+l,j}$ is the minimum between $\sum_{k=0}^l \binom{l}{k} = 2^l$ values. In particular, $D_{i,j}$ is the minimum between $2 \cdot \sum_{k=0}^{\tau} \binom{\tau}{k} - \tau = 2^{\tau+1} - \tau$ values because the set of all the paths of $D_{i,j}$ is symmetric with respect to the line from $D_{i-\tau,j-\tau}$ to $D_{i,j}$. We know that the minimum circuits consume the largest number of levels than others (equality circuit or addition circuits), and it needs $\mathcal{O}(\log k)$ levels to evaluate the minimum circuits MIN^k that compute the minimum value between k numbers, which requires $\mathcal{O}(k^2)$ homomorphic computations. Thus, one can compute a block of size- $(\tau+1)$ by evaluating the circuits with a SWHE of depth $\mathcal{O}(\log(2^{\tau+1} - \tau)) \approx \mathcal{O}(\tau)$ in

$$\sum_{k=2,2^2,\dots,2^{\tau-1}} \mathcal{O}(k^2) + \mathcal{O}((2^{\tau+1} - \tau)^2) \approx \mathcal{O}(2^{2\tau})$$

homomorphic operations. From the fact that all the blocks of size- $(\tau+1)$ can be computed diagonally while shares of the values of \mathcal{T} and \mathcal{L} have been computed, we can conclude that the edit distance algorithm can be implemented using $\mathcal{O}(2^{2\tau} \cdot \frac{nm}{\tau^2})$ homomorphic operations with a SWHE scheme of depth $\mathcal{O}(\tau \cdot (\frac{n+m}{\tau} - 1)) \approx \mathcal{O}(n+m)$ for given two encrypted sequences of lengths n and m . Hence, this optimization reduces the depth, but the entire computation increases as τ becomes larger. In particular, in the case of $n = m$, we can implement the algorithm with lesser depth circuits. The essence of the idea is formally captured in Lemma 3 below.

Lemma 3. *Let σ_j denote the elementary symmetric polynomial of degree j in x_1, x_2, \dots, x_n and $\tilde{\sigma}_j$ the binary circuit which is a conversion of σ_j by the following rules: $+$ $\mapsto \oplus$ and \cdot $\mapsto \wedge$. Also, let $\mu := \lceil \log n \rceil$. Then the addition circuits ADD^n convert the sum of n one-bit x_i ’s into a μ -bit integer, defined by $(S[1], S[2], \dots, S[\mu])$ satisfying*

$$S[i] = \bigoplus_{1 \leq j \leq n} \left(\bigoplus_{\substack{1 \leq k \leq j \\ k[i]=1}} \left[\binom{j}{k} \right]_2 \right) \wedge \tilde{\sigma}_j.$$

Proof. Denote \mathfrak{S}_n the symmetric group on the n letters and

$$X_k := \sum_{\zeta \in \mathfrak{S}_n} (x_{\zeta(1)} \cdots x_{\zeta(k)} \cdot (x_{\zeta(k+1)} + 1) \cdots (x_{\zeta(n)} + 1)).$$

Let us c_j denote a coefficient of σ_j in X_k over integers. We show that $c_j \cdot \binom{n}{j} = \binom{n-k}{j-k} \cdot \binom{n}{k}$. More precisely, the number of monomials of degree j in X_k is $c_j \cdot \binom{n}{j}$ because $\binom{n}{j}$ can be seen as the number of the monomials of σ_k .

Note that for a fixed $\zeta \in \mathfrak{S}_n$, the number of monomials of degree j in $(x_{\zeta(1)} \cdots x_{\zeta(k)} \cdot (x_{\zeta(k+1)} + 1) \cdots (x_{\zeta(n)} + 1))$ is $\binom{n-k}{j-k}$. Since the number of such polynomials is $\binom{n}{k}$, we have $c_j = \binom{j}{k}$ and $X_k = \sum c_j \sigma_j = \sum_{k \leq j \leq n} \binom{j}{k} \cdot \sigma_j$.

Now let us consider the binary circuit \tilde{X}_k , that is,

$$\tilde{X}_k = \bigoplus_{\zeta \in \mathfrak{S}_n} (x_{\zeta(1)} \wedge \cdots \wedge x_{\zeta(k)} \wedge (x_{\zeta(k+1)} \oplus 1) \wedge \cdots \wedge (x_{\zeta(n)} \oplus 1)),$$

so we have $\tilde{X}_k = \bigoplus_{k \leq j \leq n} (\left[\binom{j}{k} \right]_2 \wedge \tilde{\sigma}_j)$. Hence, we can conclude that

$$\begin{aligned} S[i] &= \bigoplus_{1 \leq k \leq n} (\tilde{X}_k \wedge k[i]) = \bigoplus_{1 \leq k \leq n} \left(\bigoplus_{k \leq j \leq n} \left[\binom{j}{k} \right]_2 \wedge \tilde{\sigma}_j \right) \wedge k[i] \\ &= \bigoplus_{\substack{1 \leq k \leq j \leq n \\ k[i]=1}} \left[\binom{j}{k} \right]_2 \wedge \tilde{\sigma}_j = \bigoplus_{1 \leq j \leq n} \left(\bigoplus_{\substack{1 \leq k \leq j \\ k[i]=1}} \left[\binom{j}{k} \right]_2 \right) \wedge \tilde{\sigma}_j. \end{aligned}$$

The first equality follows since only k values of x_1, \dots, x_n can be “1” (i.e., $\sum_{i=1}^n x_i = k$) if and only if $\tilde{X}_k = 1$. \square

The lemma implies that if we have computed “ $\bigoplus \left[\binom{j}{k} \right]_2$ ” satisfying $1 \leq k \leq j$ and $k[i] = 1$ (for $1 \leq i \leq \mu$ and $1 \leq j \leq n$), then S_i ’s are expressed in terms of the symmetric polynomials with degree no more than n . The following proposition follows from Lemma 3.

Proposition 4. *Encrypted Edit distance algorithm can be implemented on two sequences of length n over an ω -bit alphabet with a somewhat homomorphic scheme of depth*

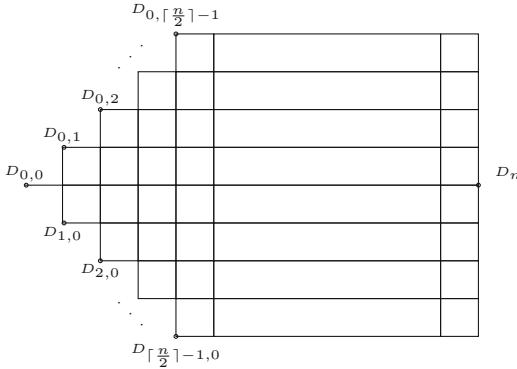
$$\lceil \log \omega \rceil + \lceil \log n \rceil + \lceil \log \left(\lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor \right) \rceil + \lceil \log(n') \rceil + 2$$

where $n' = -\lceil \frac{n}{2} \rceil - 1 + 2 \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} \binom{n}{i}$.

Proof. Let us consider a size- $(n+1)$ block. Since $D_{n,n} = D_n$ is less than n and $D_{i,0}, D_{0,i}$ are greater than $2i$, D_n can be expressed as a function of $D_{0,0}, D_{1,0}, \dots, D_{\lceil \frac{n}{2} \rceil - 1,0}, D_{0,1}, \dots, D_{0,\lceil \frac{n}{2} \rceil - 1}$, and $t_{i,j}$ ’s satisfying $|i-j| \leq \lceil \frac{n}{2} \rceil$, as shown in Fig. 4 (which means that it is enough to compute only a little part of the block).

Firstly it needs $\lceil \log \omega \rceil$ levels to compute $t_{i,j}$ ’s with the equality circuits over ω -bits. Next, from the fact that the number of the functions of D_n with respect to $D_{i,0}$ is $\binom{n}{i}$, the edit distance D_n is the minimum between $n' = -\lceil \frac{n}{2} \rceil + 2 \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} \binom{n}{i}$ values which have the following form:

$$D_{i,0} + t_{i_1,j_1} + t_{i_2,j_2} + \cdots + t_{i_{n-k},j_{n-k}} + i = 2i + t_{i_1,j_1} + t_{i_2,j_2} + \cdots + t_{i_{n-k},j_{n-k}}$$

**Fig. 4.** Grid of $(n+1)$ -block

where $1 \leq i_1 \leq i_2 \leq \dots \leq i_{n-k} \leq n$ and $1 \leq j_1 \leq j_2 \leq \dots \leq j_{n-k} \leq n$. In particular, “ $t_{1,1} + t_{2,2} + \dots + t_{n,n}$ ” has binary circuits which consume the largest levels to be evaluated, and from Lemma 3 we expect that it needs $\lceil \log n \rceil$ levels. We note that all the values to be compared are less than $n + \lceil \frac{n}{2} \rceil - 1$ and they are considered to be of $\lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor + 1$ -bit, so we have $D(\text{COM}) = \lceil \log(\lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor) \rceil + 1$. Finally, the proposition follows that

$$\begin{aligned} & D(t_{i,j}) + D(t_{1,1} + \dots + t_{n,n}) + D(\text{COM}) + D(\text{MIN}^{n'}) \\ &= (\lceil \log \omega \rceil) + (\lceil \log n \rceil) + \left(\lceil \log(\lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor) \rceil + 1 \right) + (\lceil \log(n') \rceil + 1). \square \end{aligned}$$

The result of Proposition 4 tells us that we can reduce the depth of computing edit distance to $\mathcal{O}(\log n') \approx \mathcal{O}(\log(2 \cdot 2^{\frac{n}{2}-1})) \approx \mathcal{O}(n)$. In particular, if $n = m = 8$, then the number of levels consumed by the edit distance algorithm is approximately 16.

5 Implementation and Discussions

In the following we give an estimated performance of the encrypted edit distance algorithm over DNA sequences and provide concrete timings for homomorphic evaluation of the algorithm with Shoup’s NTL library [22] and Halevi-Shoup’s HE library [13] over GMP. A complete description of this scheme is given in [8]. We may assume that $\omega = 2$ from the fact that $\Sigma = \{A, G, C, D\}$.

In our scenario, the third parties first partition their own DNA sequences into segments of length n or m . Then each of the DNA sequences is expressed in a binary representation. After that, each bit is encrypted as a different ciphertext with a homomorphic encryption scheme. For parallel computation, we use an encryption scheme with plaintext space \mathbb{Z}_2^ℓ supporting SIMD operations with ℓ slots. Then one party sends the ciphertexts which hold the ℓ segments together to a cloud. Finally, the cloud service computes the edit distances of ℓ different

sequence pairs simultaneously. The amortized time is computed as the total time of this algorithm evaluation divided by ℓ .

5.1 Estimates

In addition to the modulus switching method, there is another noise-management technique—*bootstrapping* which evaluates the decryption circuit of homomorphic encryption scheme using the decryption key. This results in a different encryption of the ciphertext with reduced noise, so the number of homomorphic operations becomes unlimited, called *fully homomorphic encryption* (FHE).

If the length of DNA sequences is large, our encrypted edit distance algorithm requires large depth. So for sufficiently long sequences, we estimate the algorithm using an FHE scheme instead of an SWHE scheme. In particular, we present the estimated performance using the batch DGHV scheme [4]. Since bootstrapping is more costly than other operations and this scheme performs a bootstrapping right after each multiplication, the number of homomorphic multiplications directly affects the total evaluation performance. We note that the edit distance algorithm in Sect. 4.3 needs many more multiplications than the one in Sect. 4.1. For these reasons, the latter is more suitable for being evaluated via FHE.

We assume that the length of DNA sequence segments is less than 100 because a single DNA sequencer can generate millions of short DNA sequences with 100–120 nucleotides. We first count the total number of homomorphic multiplications in the edit distance algorithm up to size (100, 100), which can be seen as the number of bootstrapping operations during the evaluations. Then it is multiplied by the timing for a single bootstrapping operation with their results (using the same parameters as in [4]). We present the estimates of the proposed algorithm in Table 5.

5.2 Experimental Result

Using the optimization techniques as described in Sect. 4.3, we can evaluate the edit distance algorithm homomorphically with low depth circuits for small DNA sequences. Taking 80-bits of security, we used many different parameters for several level parameters L according to the length of the two DNA sequences, that is, we chose SWHE scheme so as to support the depth which are incurred by the computations for each case. In the set up stage, we determine the parameters of a SWHE scheme and generate a secret/public key pair and the modulus switching data.

We implemented the encrypted edit distance algorithm for two sequences of length n and m . In our implementation, we use $\tau = n = m$ as mentioned before. The implementation results are described in Table 6. For example, it takes 27.5 sec. to obtain the encrypted edit distance from the two encrypted DNA sequences of length 8. This is about 45 times faster than the result of Sect. 5.1, which is expected to take 20 min. for 72-bits of security.

Table 5. Estimates of amortized timing for homomorphic edit distance computation using a FHE scheme [4]

(n,m)	Toy	Small	Medium	Large
Security	42	52	62	72
# of slots	10	37	138	531
pk size	647 KB	13.3 MB	304 MB	5.6 GB
(1, 1)	0.108 s	0.297 s	0.891 s	3.402 s
(2, 2)	1.104 s	3.046 s	9.107 s	34.776 s
(3, 3)	3.996 s	11.025 s	32.962 s	2 min 5 s
(4, 4)	7.104 s	19.600 s	58.599 s	3 min 44 s
(6, 6)	22.032 s	1 min 1 s	3 min 2 s	11 min 34 s
(8, 8)	39.168 s	1 min 48 s	5 min 23 s	20 min 34 s
(10, 10)	1 min 18 s	3 min 35 s	10 min 43 s	40 min 57 s
(20, 20)	6 min 19 s	17 min 26 s	52 min 8 s	3 h 19 min
(30, 30)	14 min 13 s	39 min 14 s	1 h 57 min	7 h 27 min
(50, 50)	46 min 30 s	2 h 8 min	6 h 23 min	1 day 24 min
(100, 100)	3 h 34 min	9 h 50 min	1 day 5 h	4 days 16 h

Table 6. Timing of an implementation of homomorphic edit distance on an Intel Xeon i7 2.3 GHz, 192 GB (80 bit security)

(n,m)	Depth of hom. enc.	Ring modulus Φ_d	ℓ	Key generation	Encryption	Total time	Amortized time
(1,1)	1	$d = 4369$	256	1.4761 s	0.1118 s	0.0693 s	0.0003 s
(2,2)	2	$d = 4369$	256	1.8358 s	0.2844 s	0.2532 s	0.0009 s
(3,3)	8	$d = 8191$	630	7.0162 s	1.7117 s	34.3091 s	0.0540 s
(4,4)	9	$d = 8191$	630	7.4489 s	2.4154 s	67.5116 s	0.1071 s
(6,6)	16	$d = 13981$	600	16.1076 s	9.9498 s	26 min 33 s	2.6555 s
(8,8)	19	$d = 15709$	682	27.5454 s	16.4524 s	5h 13 min	27.5528 s

6 Conclusion

In this paper, we proposed an algorithm to perform the edit distance algorithm on encrypted genomic sequences. More precisely, upon input two encrypted sequences of lengths n and m by a SWHE scheme, our algorithm outputs an encrypted value of their edit distance. We show that this can be done in $\mathcal{O}(nm \log(n + m))$ computations with a SWHE scheme which can homomorphically evaluate any circuit of depth $\mathcal{O}((n + m) \log(\log(n + m)))$. With our optimization technique, we can reduce the depth of computing edit distance to $\mathcal{O}(n + m)$ and the implementation shows that it takes 27.5 sec. for $n = m = 8$ using the Halevi-Shoup code [13].

Currently we could not implement our algorithm for larger parameters due to large memory requirements, but if one can manage large memory or improve the scheme to reduce the memory requirements, it is expected that the algorithm would run in one day for $n = m = 50$ when estimated based on the recent CCK+ scheme [4].

The proposed algorithm enables us to perform any sequence analysis over encrypted genomic sequences without worrying about privacy leakage. It would be very interesting to make our algorithm practical for larger parameters by improving the algorithm with the help of more efficient homomorphic encryption.

Acknowledgements. This work was supported by IT R&D program of MSIP/KEIT [No. 10047212] and the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1010) supervised by the NIPA (National IT Industry Promotion Agency). The authors would like to thank the anonymous reviewers of WAHC 2015 for their helpful comments.

References

1. Atallah, M.J., Kerschbaum, F., Du, W.: Secure and private sequence comparisons. In: WPES, pp. 39–44 (2003)
2. Ayday, E., Hubaux, J.-P., Raisaro, J.L., Rougemont, J.: Protecting and evaluating genomic privacy in medical tests and personalized medicine. In: WPES, pp. 95–106 (2013)
3. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325 (2012)
4. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
5. Cristofaro, E.D., Faber, S., Tsudik, G.: Secure genomic testing with size- and position-hiding private substring matching. In: WPES, pp. 107–117 (2013)
6. The European Bioinformatics Institute. <http://www.ebi.ac.uk>
7. Erlich, Y., Narayanan, A.: Routes for breaching and protecting genetic privacy (2013). [arXiv:1310.3197](https://arxiv.org/abs/1310.3197)
8. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
9. Gymrek, M., McGuire, A.L., Golan, D., Halperin, E., Erlich, Y.: Identifying personal genomes by surname inference. *Science* **339**, 321–324 (2013)
10. Humbert, M., Ayday, E., Hubaux, J.-P., Telenti, A.: Addressing the concerns of the lacks family: Quantification of kin genomic privacy. In: CCSW Secure Pattern Matching using Somewhat Homomorphic Encryption, pp. 1141–1152. ACM (2013)
11. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: Proceedings of the 20th USENIX Security Symposium, pp. 35–50 (2011)
12. HapMap (2007). <http://www.hapmap.org/>

13. Halev, S., Shoup, V.: Design and implementation of a homomorphic-encryption library. Technical report, IBM Technical report (2013)
14. Jha, S., Kruger, L., Shmatikov, V.: Towards practical privacy for genomic computation. In: IEEE Symposium on Security and Privacy, pp. 216–230 (2008)
15. Kantarcioglu, M., Jiang, W., Liu, Y., Malin, B.: A cryptographic approach to securely share and query genomic sequences. *IEEE Trans. Inf. Technol. Biomed.* **12**(5), 606–617 (2008)
16. 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)
17. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical?. In: CCSW, pp. 113–124. ACM (2011)
18. Lindell, Y., Pinkas, B.: A proof of Yao’s protocol for secure two-party computation (2004). <http://eprint.iacr.org/2004/175>
19. Malin, B., Sweeney, L.: Inferring genotype from clinical phenotype through a knowledge based algorithm. In: Pac. Symp. Biocomput. 41–52 (2002)
20. Malin, B., Sweeney, L.: How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *J. Biomed. Inform.* **37**(3), 571–588 (2004)
21. Personal Genome Project. <http://www.personalgenomes.org/community.html>
22. Shoup, V.: NTL: a library for doing number theory (2009). <http://www.shoup.net/ntl>
23. Sweeney, L., Abu, A., Winn, J.: Identifying Participants in the Personal Genome Project by Name. In: Harvard University. Data Privacy Lab. White Paper 1021–1 (2013)
24. Stranger Visions (2012). <http://deweyhagborg.com/strangervisions>
25. Wagner, R.A., Fischer, M.J.: The string to string correction problem. *J. ACM* **21**(1), 168–173 (1974)
26. Yao, A.: How to generate and exchange secrets. In: Ostrovsky, R. (ed.) FOCS, pp. 162–167 (1986)
27. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Secure pattern matching using somewhat homomorphic encryption. In: CCSW, pp. 65–76. ACM (2013)

HEtest: A Homomorphic Encryption Testing Framework

Mayank Varia¹, Sophia Yakoubov¹, and Yang Yang^{2(✉)}

¹ MIT Lincoln Laboratory, Lexington, MA, USA

{mayank.varia,sophia.yakoubov}@ll.mit.edu

² Blizzard Entertainment, Irvine, CA, USA

y4n9@alum.mit.edu

Abstract. In this work, we present a generic open-source software framework that can evaluate the correctness and performance of homomorphic encryption software. Our framework, called `HEtest`, automates the entire process of a test: generation of data for testing (such as circuits and inputs), execution of a test, comparison of performance to an insecure baseline, statistical analysis of the test results, and production of a LaTeX report. To illustrate the capability of our framework, we present a case study of our analysis of the open-source HElib homomorphic encryption software. We stress though that `HEtest` is written in a modular fashion, so it can easily be adapted to test any homomorphic encryption software.

1 Introduction

Homomorphic encryption is a cryptographic primitive that enables computation directly on encrypted data. This technology has the potential to change the way that we protect arbitrary computation on the cloud. Moreover, it may be judiciously applied to special-purpose problems such as database searches in order to develop usable technologies [1] with stronger security guarantees than were possible before [4, 18, 20].

The last five years have seen substantial research into the design of homomorphic encryption algorithms [2, 3, 5, 7, 9, 11, 23, 24]. Some of these algorithms have been implemented in software [8, 10, 12, 13]. Evaluations of these software implementations allow the research community to determine the applications that could benefit most from targeted use of homomorphic encryption technology. However, prior evaluations of homomorphic encryption software were tedious to conduct and *ad hoc* in nature. This resulted in evaluations that cannot be directly compared because the data were produced on different platforms. Additionally,

¹ Y. Yang—Work performed while at MIT Lincoln Laboratory.

This work is sponsored by the Intelligence Advanced Research Projects Activity under Air Force Contract FA8721-05-C-002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

while tests were repeatable in principle, the *ad hoc* nature of prior evaluation software made it challenging to reproduce others' test results.

Our main contribution is to provide a generic, open-source framework for the testing of homomorphic encryption schemes. In this paper, we describe the design of our test framework and our use of this framework to evaluate portions of the HElib software package [12, 13].

Homomorphic Encryption. The goal of *fully homomorphic encryption* (FHE) research is to design an encryption scheme that is purposely malleable in a specific way to enable computation on encrypted data. More specifically, an FHE scheme has a special operation `Evaluate` that takes a circuit representation C of a program and a series of ciphertexts $c_i = \text{Enc}(m_i)$ and returns an encryption of $C(m_1, \dots, m_k)$.

Due to their number-theoretic properties, many public-key encryption schemes are naturally homomorphic with respect to a single operation like addition or multiplication [6, 17, 19, 22]. An intriguing question, initially posed in 1978 by Rivest et al. [21], is whether there exists an encryption scheme that simultaneously permits the evaluation of *both* addition and multiplication on ciphertexts. Since these two operations constitute a logically complete set of operations, the vision described above would follow.

In 2009, the seminal work of Gentry [7] affirmatively answered this long-standing question by demonstrating an encryption scheme based on ideal lattices that exploits the ring structure of the ciphertext space to provide `Evaluate` operations for both addition and multiplication. Gentry split the FHE problem into two components: the design of a *somewhat homomorphic encryption scheme* (SWHE) that permitted a limited number of `Evaluate` operations and the insight of a *bootstrapping* algorithm that achieved fully homomorphic encryption through the use of multiple applications of SWHE. A slight tweak of this initial scheme was implemented in [8].

After Gentry's initial work, many cryptographers have designed new FHE algorithms [2, 3, 5, 9, 11, 23, 24] that make substantial improvements along several dimensions:

- Improving the performance of a single addition or multiplication `Evaluate` operation,
- Increasing the number of `Evaluate` operations possible in a SWHE scheme before bootstrapping,
- Batching multiple plaintext bits into one ciphertext whose bits are operated on in parallel, and
- Basing the cryptography upon weaker, more accepted number-theoretic assumptions.

Some of these improved algorithms have been implemented in software [10, 12, 13]. However, the FHE community lacks a simple benchmarking tool that enables simple comparisons between these homomorphic encryption schemes and with naïve unprotected computation.

Our Software. We have designed and developed a software program for evaluating homomorphic encryption schemes called **HEtest** [15, 26]. Our code has been open-sourced for use under a BSD license [16] and is available for download at <https://www.ll.mit.edu/mission/cybersec/softwaretools/hetest/hetest.html>. The **HEtest** software package comprises four components:

1. A *circuit generator* that can create configurable instances of boolean circuits. Customizable options include circuit depth and the desired distribution of gates in a circuit.
2. A *baseline* that parses and evaluates the circuits as quickly as possible without homomorphic encryption, for comparison purposes.
3. A *test harness* that interfaces with any homomorphic encryption software through a communication protocol. The harness executes *test scripts* that repeatedly call for key generation, circuit ingestion (using the circuits generated above), encryption, homomorphic evaluation, and decryption. During a test, the harness captures and logs metrics pertaining to the correctness and performance of the homomorphic encryption software.
4. A *report generator* that (with no human input) analyses the test harness' logs and produces a LaTeX report with tables and graphs that summarize the correctness and performance results (both in absolute terms and relative to the baseline).

The circuit and report generators are written in Python, whereas the baseline and test harness, which are performance-critical, are written in C++.

Our Testing. We used our **HEtest** software to evaluate submissions to the Security and Privacy Assurance Research (SPAR) program. The SPAR program was developed in 2010 by the Intelligence Advanced Research Projects Activity (IARPA). Its objective was to design and build new privacy-preserving data searching technologies that are fast and expressive enough to use in practice.

The SPAR program comprised nine research teams who worked on three separate problems, two of which were about the design and implementation of privacy-preserving database or publish-subscribe schemes [4, 14, 20, 25]. The final component of the project focused on the design of homomorphic encryption schemes, with the aim of producing an efficient SWHE building block that could be used in the design of privacy-preserving search algorithms with strong security guarantees. There were two performers involved in the homomorphic encryption component of SPAR: the IBM Thomas J. Watson Research Center (hereafter referred to as IBM) and Stealth Software Technologies, Inc (hereafter referred to as Stealth).

In this paper, we provide a case study for the use of the **HEtest** framework by describing our evaluation of the IBM submission to the SPAR project, which is largely based on the open-source HElib implementation [12, 13]. Thus, the results of our case study can easily be reproduced by interested readers. We wish to emphasize that our discussion of HElib in this paper is merely as a case study: we used **HEtest** to evaluate Stealth's software as well, and all of the code in **HEtest** is written in a modular, extensible fashion, so it can easily be extended

to test other homomorphic encryption software with only minor changes to the data generation and baseline code if support is needed for different types of circuits.

Organization. The rest of this paper is organized as follows. In Sect. 2, we provide a definition for homomorphic encryption and a brief overview of HElib [12, 13]. In Sect. 3, we describe the process by which `HEtest` generates and stores data. Section 4 describes the test execution framework in `HEtest`. Section 5 explains `HEtest`'s automated data analysis and report generation. Finally, Sect. 6 provides a case study of the use of our framework to study IBM's HElib software.

2 Overview of Homomorphic Encryption and HElib

A basic public-key encryption scheme has the three algorithms `KeyGen`, `Encrypt`, and `Decrypt`. `KeyGen` is a randomized algorithm that inputs a security parameter λ and outputs a public/secret key pair (pk, sk) . `Encrypt` is a randomized algorithm that takes a public key pk and a plaintext message m from the plaintext space \mathcal{P} and outputs a ciphertext c from the ciphertext space \mathcal{C} . Finally, `Decrypt` is an algorithm that takes as input a secret key sk and a ciphertext c and outputs a plaintext message $m \in \mathcal{P}$. The computational complexity of these algorithms must be polynomial in λ .

A homomorphic encryption scheme has an additional algorithm `Evaluate`, which takes as input a public key pk , a function represented as a circuit C , and a vector of ciphertexts $\mathbf{c} = (c_1, \dots, c_w)$ where c_i is the encryption of m_i . It outputs a ciphertext c' that is an encryption under pk of $C(m_1, \dots, m_w)$, the result of evaluating the circuit C using m_1, \dots, m_w as inputs. The scheme satisfies the homomorphic property that for all (pk, sk) , circuits C , and $c_i = \text{Encrypt}(pk, m_i)$,

$$\text{Decrypt}(sk, \text{Evaluate}(pk, C, c_1, \dots, c_w)) = C(m_1, \dots, m_w).$$

In recent years, one construction of a homomorphic encryption scheme based on the ring-LWE assumption by Brakerski, Gentry, and Vaikuntanathan [2] has shown promise of becoming “somewhat practical.” In this scheme, plaintext bits are represented as coefficients of a polynomial in the ring $\mathbb{F}_p[x]/(f(x))$. Gentry, Halevi, and Smart [11] design a variant of the BGV scheme in which $p = 2$ and $f(x)$ is chosen to be the n -th cyclotomic polynomial $\Phi_n(x)$. IBM's HElib software is an implementation of this cryptosystem, with further optimizations in ciphertext packing or “batching” [24]. Specifically, if the polynomial ring $\Phi_n(x)$ can be factored modulo 2 into ℓ irreducible factors, then there are ℓ “slots” in which one can encode a plaintext bit by application of the Chinese Remainder Theorem for polynomials. Using this construction, addition and multiplication in the polynomial ring $\mathbb{F}_p[x]/(f(x))$ correspond to element-wise addition and multiplication in the vector of slots, giving rise to single instruction multiple data (SIMD) style operations.

If c is the smallest integer such that n divides $p^c - 1$, then $\Phi_n(x)$ factors into $\ell = \phi(n)/c$ irreducible polynomials modulo p , where $\phi(\cdot)$ denotes Euler's totient function. In order to maximize ℓ , one needs to choose an n that minimizes c .

Table 1. Depth of each gate type, as defined by IBM.

Gate	Depth (d)
MULT	1
MULTconst	0.5
ADD	0.1
ADDconst	0
SELECT	0.6
ROTATE ^a	0.25 to 0.75

^aThe depth of a ROTATE gate depends on ℓ , the number of plaintexts packed or “batched” into a ciphertext.

However, n is also constrained by the choice of security parameter λ and the maximum circuit depth d .

In the HElib cryptosystem, the parameter n was set between 4500 and 45000 for $\lambda = 80$ bits of security and $d \in [4, 24]$. These settings yielded batch widths of approximately $\ell \in [256, 1285]$. Their scheme produced ciphertexts with bit-size asymptotically equal to $O(\phi(n) \cdot d \cdot \log(\lambda))$. Since ciphertext blow-up represents an enormous cost of computing data homomorphically, packing multiple independent plaintext bits into a single ciphertext is a substantial improvement to efficiency.

In HElib, the circuit depth d has a special meaning because the various SIMD operations introduce different amounts of noise to the ciphertext. For instance, adding two ciphertexts increases the noise in the resulting ciphertext linearly while multiplying two ciphertexts increases the noise quadratically. Consequently, MULT increases the circuit depth substantially more than ADD. Table 1 lists the contributions to circuit depth made by the six gate types supported by HElib. We stress that a gate’s depth is different than its *level*: the minimum number of gates between it and the input wires.

3 Test Data

In this section, we describe the process of generating circuits and corresponding inputs on which to test homomorphic encryption software such as IBM’s HElib or Stealth’s software. We stress that our tool can generate a diverse set of circuits using various gate types. In this section, we describe the generation of circuits that IBM supported: deep circuits consisting of arithmetic SIMD gates with fan-in 2. However, we also used HEtest to evaluate Stealth’s software on wide, shallow Boolean circuits with large fan-in.

We developed a Python script that generated circuit descriptions and inputs based on configurable circuit parameters. Circuit descriptions were output in ASCII-format and stored as text files, later to be parsed by the test harness

described in Sect. 4.1 and the performers’ software. In Sect. 3.1, we discuss the input parameters to our data generation system. In Sect. 3.2, we discuss the process by which a circuit and a corresponding input are constructed, given those inputs. In Sect. 3.3, we discuss the format in which the test harness expects the circuit and input data, and in Sect. 3.4, we discuss the format in which our analysis tools discussed in Sect. 5 expect the data.

3.1 Generation Parameters

The parameters for circuit generation include desired circuit width w (i.e. number of input wires), circuit depth d , batch size ℓ , the security parameter λ , and the distribution of gate types that the circuit comprises. Optionally, a random seed could be specified to reliably reproduce data. If the seed is omitted, a random one is chosen at runtime. All parameters were contained in a configuration file that the script read.

Most of our tests were run on circuits consisting of a uniformly selected set of gates over the six types shown in Table 1; we refer to these tests as ‘mixed.’ However, it was also very useful to be able to produce circuits consisting entirely of one of the six gate types, so as to be able to compare the relative efficiency of HElib’s evaluation of these gate types. Note that for circuits composed entirely of one gate type, a different notion of depth was needed because some gate types do not contribute to depth d . Therefore, for such single gate type circuits we used the number of levels `num_levels` in place of depth, referring to the length of the longest path from the output gate to any of the input wires.

3.2 Circuit and Input Generation

Circuits are generated starting with the input wires and ending with the output gate. A total of w input wires were created, and for each subsequent level of the circuit, w gates were created for which either one or two inputs (depending on the gate type) are randomly chosen from amongst the gates and wires of the two levels above it. If the test type was ‘mixed,’ generation went on until all of the gates at the last level had depth greater than d ; then, a random gate from the set of gates with the correct depth d was chosen to be the output gate. If the test type was not ‘mixed,’ generation went on until `num_levels` levels had been generated, and a random gate from the `num_levels`th level was chosen to be the output gate. Once the output gate was chosen, all gates and wires that did not contribute to the output were discarded. For each desired input, w binary strings of length ℓ were generated.

3.3 Test Suite Representation

Once the test data was generated, it had to be stored in a way that was easily accessible to both the test harness and the prototype. Each generated security parameter, circuit, and input was stored in a separate text file. Inputs were

naturally represented as lists of binary strings; for instance, here is an example of an input for a circuit with $w = 4$ input wires, each expecting an input of size $\ell = 5$: “[11010,01011,01010,11011].”

Our syntax for describing circuits is somewhat more involved, and we illustrate it here with an example in both written and graphical form in Fig. 1. The first line of Fig. 1 specifies the number of input wires (w), the depth of the circuit (d) as defined by IBM, and the batch size (ℓ). Following this header, all gates are listed ordered by their level (not depth d). This guarantees that all inputs to a gate are defined before the gate is defined. (Note, however, that gates on the same level appear in arbitrary order.) Each gate is identified by a string containing the character ‘G’ followed by a unique id. Note that not all gate ids between 1 and the maximum gate id are represented; this is because not all gates end up contributing to the output gate, and those that do not get dropped. The input wires are indexed by a string containing the character ‘W’ followed by a unique id in $0 \dots w - 1$. Following the gate id, on the same line, is the gate type and a list of the gate’s inputs. These can be pointers to input wires, other gates, or constants if the gate type requires a constant input. Note that the wires are not defined separately in the circuit description; they appear only as inputs to gates. Any constants will always be the last of the gate’s inputs.

```

W=4,D=4,L=5
G4:LMULconst(W2,01101)
G6:LMUL(W2,W0)
G5:LROTATE(W2,4)
G8:LMUL(G6,W2)
G9:LMULconst(G5,00101)
G7:LADD(G4,W0)
G13:LMULconst(G8,00000)
G11:LMULconst(G8,11010)
G14:LSELECT(G11,G7,11001)
G16:LSELECT(G13,G9,00001)
G18:LSELECT(G14,G13,11110)
G19:LADDconst(G16,10000)
G22:LMULconst(G19,01000)
G25:LADD(G22,G18)
G26:LMULconst(G25,11010)

```

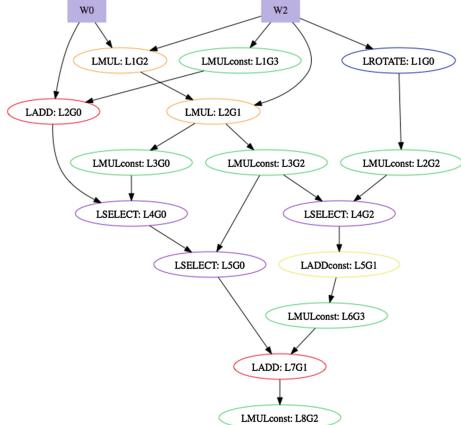


Fig. 1. The syntax of a generated circuit (left) and a graphical illustration of the same circuit (right). In the illustration, gates are labeled first by their level (e.g. ‘L4’) and then by their id within that level (e.g. ‘G2’).

Once the security parameter, circuit and input files were created, a single *test script* file corresponding to the test suite was produced. It contained the paths to the files that stored each data object, in the order in which they were meant to be sent to the performer binaries. This test script facilitates the test execution process. Finally, in order to have a common repository for all test artifacts, the parameters of each circuit and input were stored in a SQLite database that is described in Sect. 3.4.

3.4 SQLite Database

HEtest uses a SQLite database as a central repository for test information. This database served as the “glue” that enabled integration of the components of our test framework and the automation of our entire test process. We use a SQLite database because it is lightweight, SQL-based, and easy to share and back up.

Our SQLite database is built during the data generation process, and it is initially populated with some descriptive information about the circuits and inputs such as the circuit depth, the number of input wires, and the number of gates of each type present. Baseline and performer test results are later automatically added upon execution of a test, as detailed in Sect. 4. The SQLite database is used during automatic generation of a report characterizing the correctness and performance of the homomorphic encryption prototype, as described in Sect. 5. Because of all the circuit- and input-specific information we store, our report can correlate performance with specific circuit parameters, making for a very detailed analysis. We are also able to add new metrics at any time, without having to repeat the entire testing process, because the SQLite database contains all of the necessary information.

4 The Test Framework

The design of our test framework was motivated by three goals. First, we wanted to assess the performance of homomorphic encryption schemes by measuring the duration of key generation, encryption, decryption, and homomorphic evaluation. Second, we wanted to characterize the overhead of privacy assurance by comparing the system that uses homomorphic encryption to one that offers no security. Finally, we wanted to design a test harness that could be used to evaluate arbitrary homomorphic encryption schemes (such as those from IBM and Stealth) in a black-box manner.

The homomorphic encryption software being instrumented by our test framework comprised two processes: a *server* that performed homomorphic evaluation and a *client* that performed key generation, encryption, and decryption. They are collectively called the *system under test*, or SUT.

4.1 The Test Harness

The *test harness*, a program that we developed in C++, spawned the client and server processes of the SUT. It communicated with the SUT through the client’s and server’s standard input and output streams. After both processes were properly initialized, the test harness called on them to repeatedly perform key generation, encryption, circuit ingestion, homomorphic evaluation, and decryption. A configuration file, read by the test harness on start up, specified the location of files containing the security parameters, plaintext inputs, and circuits to be used.

In the key generation step, the test harness sends the value of the security parameter to the client and receives a public key. In the circuit ingestion step,

the public key and circuit description are sent to the server. When the server finishes parsing the circuit description and is ready to accept inputs, it returns a **READY** message to the test harness. In the encryption step, the test harness sends a series of plaintext messages to the client and receives their ciphertexts. Next, in the homomorphic evaluation step, the ciphertexts are sent to the server. The server evaluates the circuit using the ciphertexts as input and returns a ciphertext representing the output to the circuit. Finally, in the decryption step, the test harness forwards the ciphertext returned by the server to the client. The client decrypts the ciphertext and returns the plaintext message. Figure 2 illustrates these steps.

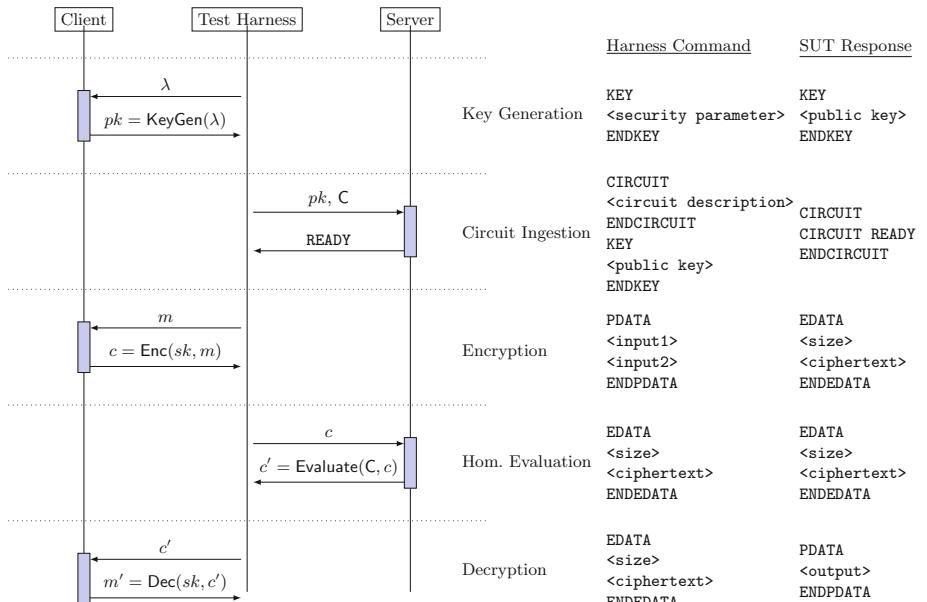


Fig. 2. Sequence diagram showing data flows during a test (left) and the actual packets sent between the test harness and SUT client/server (right). Angle brackets denote variables to be replaced by actual values.

Communication between the test harness and the SUT was dictated by a simple packet-based communication protocol that we developed. A packet consisted of a header, either ASCII-encoded plaintext data or binary-encoded ciphertext data, and a footer. These components were delimited by linefeeds. Security parameters, public keys, plaintext messages, and circuit descriptions were encoded in ASCII while ciphertexts were encoded in binary (with a prefix indicating the size of the binary payload in bytes). Our testing framework assumed that the client and server only communicate through the test harness interface (as shown in Fig. 2), never with each other directly.

In each of the steps described above, the duration of the cryptographic operation was measured from when the test harness wrote the first byte of command packet to when last byte of the response packet was read. Consequently, unavoidable communication overhead such as writing data to pipes were captured; however, the latency of these calls were negligible (about 10^{-4} seconds) compared to those of cryptographic operations. Our test harness captured the following metrics:

1. **Evaluation Accuracy:** The fraction of circuit evaluations that are correct, i.e. the decrypted result returned by the SUT is equal to the result of directly evaluating the circuit on the plaintext inputs.
2. **Key Generation Time:** A measure of how long it takes for the client to generate cryptographic keys.
3. **Key Size:** The size of public keys generated by the client prototype
4. **Ingestion Time:** A measure of how long it takes for the server to prepare a circuit for evaluation.
5. **Encryption Time:** A measure of how long it takes for the client to encrypt a plaintext message.
6. **Ciphertext Size:** The average size of ciphertext per bit of plaintext input.
7. **Evaluation Time:** A measure of how long it takes for the server to evaluate a circuit.
8. **Decryption Time:** A measure of how long it takes the client to decrypt a ciphertext.
9. **Total elapsed Time:** The sum of the encryption, evaluation, and decryption times.

4.2 The Baseline

In order to characterize performance in a setting without privacy assurance, we provided a *baseline*: a client-server implementation of a circuit evaluator that operates on plaintext inputs. The client offered stubbed implementations of key generation, encryption, and decryption in order to adhere to the communication protocol. They returned properly-formed response packets. The server supported two operations: circuit ingestion and direct evaluation on plaintext inputs.

Like the test harness, we developed the baseline system in C/C++. It was reasonably efficient, with optimizations that would normally be found in circuit evaluation programs. Most notably, it short-circuits gate operations when possible (e.g., an AND gate returns false as soon as one of its inputs is determined to be false). In addition, we provided an API for defining new gate types should the need arise.

In the circuit ingestion step, the baseline uses a scanner and parser to read textual circuit descriptions and construct circuits. A scanner, oftentimes called a tokenizer, is a program that recognizes lexical patterns in text. Any circuit description output by our circuit generator tool that we described in Sect. 3 can be read and tokenized by the scanner. These tokens are subsequently fed into a parser, which constructs circuit gates and eventually builds a circuit.

We recognized that implementing scanners and parsers for circuit descriptions is tedious, error-prone, and non-extensible. As a result, we used two tools, Flex and Lemon, to programmatically generate these software components. Flex is a scanner generator – given a set of rules (i.e. mappings between regular expressions and tokens), it outputs C source code, which when compiled, produces a scanner. The rules that we used to parse IBM-style circuits are found in the `ibm-scanner.l` file in our open-source repository. Lemon is a parser generator – given a context-free grammar, it produces C source code, which when compiled, produces a parser. The grammar for IBM-style circuits is defined in the `ibm-parser.y` file. To extend our baseline to evaluate a new gate type, one would need to add an additional rule, define a new token type, and provide a C++ implementation of the gate that extends the base gate type.

5 Report Generation

After executing a test, the final step in the HEtest chain is the production of a concise report that presents the correctness and performance results in such a way that a human can quickly draw intelligent conclusions about the prototype performance.

In the initial version of our software, this process was mostly performed manually with the aid of a few analysis scripts that produced the graphs and tables we desired. However, the addition of any new data necessitated a repetition of the entire process, which was tedious and time-consuming.

To simplify this task, we developed a tool that automatically generated a detailed report describing the performance and correctness of the prototype. In addition to giving us a summary of the system’s performance within seconds after the completion of a test, this tool allowed us to identify odd or unexpected behaviors exhibited by the system in near real-time without having to manually search through test data.

The report generator read from a centralized SQLite database containing all of the timing and correctness data, as well as all of the parameters of the tests. It automatically performed analyses of the correctness of the homomorphic encryption software under test (i.e., whether its outputs agree with those from the baseline), and it also determined the dependency of the latency on various factors such as input size, batch size, and circuit depth. It characterized the latency both in absolute terms and relative to the baseline described in Sect. 4.2.

In Sect. 6, we display the power of the report generator by showing the results of an execution of HEtest on the IBM HELib software. Note that the formats of graphs and tables were stored in easily-updated template files, so the tool could easily be extended to produce a new type of graph or table if desired.

6 Experimental Results

In this section, we present some of the auto-generated analyses that we performed over the HELib test results. All of the statistical analysis and graphs in this

section, as well as much of the expository text, were automatically produced by our report generator tool.

We stress that our use of HElib is mainly as a case study. While we do believe that the data about HElib in this section will be of interest to some readers, we are including this data principally to demonstrate the capacity of our `HEtest` tool.

6.1 Experimental Setup

We ran the test harness and performers' system on a Dell PowerEdge R710 server machine with two Intel Xeon X5650 processors and 96 GB of RAM. All software was run on the 64-bit Ubuntu 12.04 LTS Linux distribution.

Throughout this section, all times will be presented in units of seconds and all sizes will be presented in units of bytes; for brevity, we will often omit a statement of units. Also in the interest of brevity, we only present here a subset of our results. For instance, we describe the results for security parameter $k = 80$, which provides 80-bits of security, but omit the results for $k = 128$.

6.2 Real-World Applicability

Before presenting our results, we wish to issue a warning that the data from `HEtest` does not easily translate into intuition about the performance of HElib (or any existing homomorphic encryption scheme) in real-world applications. This is because the optimal representation of the functions in real-world applications is rarely as a circuit; there is almost always conditional logic involved, and no existing homomorphic encryption scheme supports such logic directly. In order to use HElib to securely evaluate a real-world function, the function would first have to be re-written as a circuit, which would almost surely cause a significant slow-down even if it is computed in the clear. This complication motivates the creation of our baseline in Sect. 4.2: it isolates the slow-down in computing due to homomorphic encryption from that caused by the (inefficient) circuit representation.

Today, HElib can be used for specific small components of real-world applications that can be naturally represented as a circuit; we hope that the analyses in this section will illuminate the costs associated with this.

6.3 Parameters Tested

Table 2 describes the parameter values on which we tested HElib. Note that we only include values for $k = 80$; the values of d , ℓ and w are different for $k = 128$. We also tested two additional 'large' circuit settings, with $(\ell = 682, d = 24, w = 200)$ and $(\ell = 1285, d = 60, w = 50)$. For each of the above combinations, we generated two circuits, with five inputs each. Additionally, for each of the six gate types, we generated 20 circuits composed entirely of that gate type, with 5 inputs each. These circuits each had 5 levels, and $w = 100$. Ten circuits of each gate type had $\ell = 6$, and ten had $\ell = 42$.

Table 2. Parameters tested for $k = 80$, based on batch width ℓ , circuit depth d , and maximum number of inputs w .

Values of ℓ	Values of d	Values of w
378	{6, 7}	{4, 10, 20, 50, 100, 1000}
630	12	{4, 10, 20, 50, 100, 200}
600	18	{4, 10, 20, 50, 100}
682	{21, 24}	{4, 10, 20, 50, 100}

6.4 Overview of Results

We tested HElib for correctness and performance. Our report generator tool automatically determined that HElib had perfect correctness during the test: for all 1582 circuit/input pairs tested, the outputs from HElib matched those from our baseline. Additionally, the average ratio between the total elapsed time of HElib and an insecure baseline was approximately 52,600.

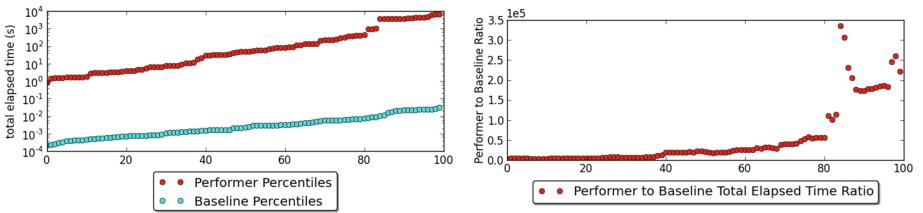


Fig. 3. Total elapsed time percentiles for HElib and the baseline separately (left) and their ratio (right)

To provide a more detailed analysis of the total elapsed time for HElib and our insecure baseline, the report generator created the two graphs in Fig. 3. The left graph in the figure shows a histogram of the total elapsed time for HElib and our baseline over the 1582 circuit/input pairs. For legibility, the data are grouped into percentiles: the graph shows the time for the fastest 1% of circuit/input pairs tested (i.e., the 16th fastest test out of 1582), then for the next 1% of tests, and so on. The right graph shows the ratio between HElib and the baseline; in other words, it is the quotient between the two curves on the left graph. These figures demonstrate that the overhead of homomorphic encryption grows as the circuits evaluated become deeper and more complex.

6.5 Key Generation

We found that key generation time and key size were highly correlated with the circuit depth d and batch width ℓ , but were not correlated with the number of

inputs w . Note that the combinations of d , ℓ , w , and k were selected jointly for our test at IBM’s request, and thus the relationships between the variables may be more complex than presented here.

Table 3. Key sizes, in bytes

Count	25
Mean	$2.02 \cdot 10^8$
Std Dev	$2.75 \cdot 10^8$
Min	$9.28 \cdot 10^5$
Max	$7.08 \cdot 10^8$

Figure 4 presents a profile of key generation time varying as a function of circuit depth d and batch width ℓ . In the ranges that we tested, both generation time and key size are linear in ℓ and quadratic in d . Our best-fit model of key generation time is:

$$t = 0.07d^2 - 1.21d + 0.013\ell + 1.4,$$

with an r^2 value of 1.0. Table 3 also provides descriptive statistics for generated key size.

6.6 Circuit Ingestion

Overall, HElib’s circuit ingestion is very fast, and is negligible compared to the time taken for other parts of the scheme. We believe that the ingestion times we observed depended primarily on the simple task of parsing rather than on any complexity of the performer’s scheme. Circuit description sizes varied in the kilobyte to low megabyte range.

Some basic statistics about circuit ingestion latency are provided in Table 4. Our analysis reveals that ingestion time was mildly correlated with ℓ , but we attribute this to the fact that the bit representation of a circuit description in our format increases with ℓ because gates that have a constant parameter (such as “add a constant to the input” or “multiply by a constant”) require ℓ bits to describe.

Table 4. Circuit ingestion latency (left), encryption latency (middle), and decryption latency (right), in seconds

Ingestion		Encryption		Decryption	
Count	81	Count	999	Count	999
Mean	0.009	Mean	0.033	Mean	0.116
Std Dev	0.017	Std Dev	0.304	Std Dev	0.36
Min	0.0	Min	0.0	Min	0.0
Max	0.119	Max	5.694	Max	5.091

6.7 Encryption and Decryption

The total encryption time was very fast, with our data (displayed in Table 4) showing that encryption took just 33 milliseconds on average. However, the collected data are contaminated because IBM’s software did not adhere to our test harness’ communication protocol, so our encryption timer erroneously included network transmission time.

Decryption time is also fast in HElib. While it can take up to 5 seconds for the largest circuits, even this amount of time is negligible compared to the hours such circuits would take for homomorphic evaluation. See Table 4 for detailed statistics.

Finally, **HEtest** captures ciphertext sizes for both “fresh” ciphertexts (i.e., after encryption and before evaluation) and “evaluated” ciphertexts (i.e., after evaluation and before decryption). A summary of these data are shown in Table 5.

Table 5. Average sizes of fresh and evaluated ciphertexts, as a function of batch size

ℓ	Count	Fresh CT size	Evaluated CT size
6	306	5.41 KB	5.65 KB
42	300	203 KB	212 KB
378	121	593 KB	864 KB
600	51	3.60 MB	3.59 MB
630	60	1.66 MB	1.81 MB
682	151	4.16 MB	6.04 MB
1285	10	42.0 MB	46.8 MB

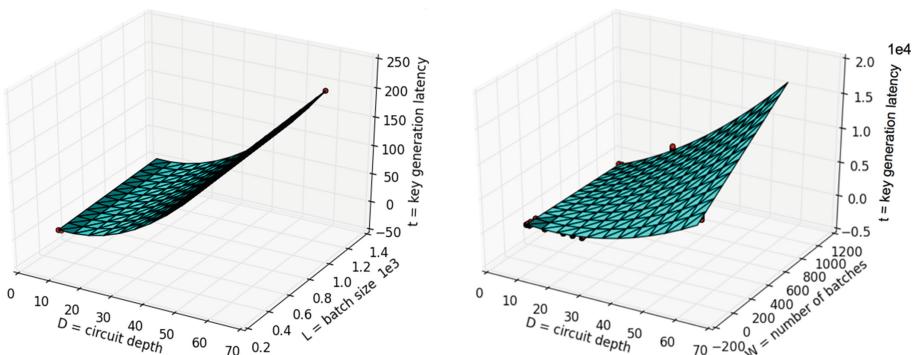


Fig. 4. Key generation time (left) and homomorphic evaluation time (right), in seconds

6.8 Homomorphic Evaluation

Evaluation time was highly correlated with d and w . It also showed a small correlation with ℓ , but the vast majority of this correlation can be explained as a result of the parameters being selected jointly. The best-fit model for $k = 80$ is

$$t = 2.77d^2 - 74.6d - 1.43w + 0.215wd + 403,$$

with an r^2 value of 0.991. Figure 4 shows a graph of evaluation time.

6.9 Evaluation Time by Gate Type

Finally, we ran several circuits through the performers' system whereby all gates were of the same type. These tests give us an indication of the time required to compute a single gate of each of the various types supported by HElib. Our results are shown in Table 6. Here, our measurements are averaged across all levels of a circuit. If a homomorphic encryption scheme has the property that the performance of gates varies substantially by level, this analysis would not be useful. Due to the special-purpose nature of single gate type tests, note that these data are not included in any of the analyses done in the prior sections.

Table 6. Evaluation time per gate, in seconds

Gate Type	Count	Mean	Std Dev	Min	Max
ADD	101	$2.04 \cdot 10^{-4}$	$1.99 \cdot 10^{-4}$	$1.10 \cdot 10^{-5}$	$6.18 \cdot 10^{-4}$
ADDconst	101	$1.85 \cdot 10^{-4}$	$1.75 \cdot 10^{-3}$	$6.00 \cdot 10^{-5}$	$4.43 \cdot 10^{-3}$
MULT	101	$1.45 \cdot 10^{-2}$	$1.39 \cdot 10^{-2}$	$4.76 \cdot 10^{-4}$	$3.00 \cdot 10^{-2}$
MULTconst	101	$1.92 \cdot 10^{-3}$	$1.82 \cdot 10^{-3}$	$7.60 \cdot 10^{-5}$	$5.19 \cdot 10^{-3}$
ROTATE	101	$1.19 \cdot 10^{-2}$	$1.14 \cdot 10^{-2}$	$2.07 \cdot 10^{-4}$	$2.59 \cdot 10^{-2}$
SELECT	101	$1.72 \cdot 10^{-3}$	$1.62 \cdot 10^{-3}$	$6.10 \cdot 10^{-5}$	$3.60 \cdot 10^{-3}$

7 Conclusion

In this work, we built a comprehensive framework for the test and evaluation of homomorphic encryption software with a focus on generalizability, test automation, and integration of test components. We presented a case study application of our `HEtest` software to the IBM HElib software. We stress though that our test framework can be easily adapted to test any other homomorphic encryption software.

We have open-sourced `HEtest` under a BSD license [16]. We encourage interested readers to download our code at <https://www.ll.mit.edu/mission/cybersec/softwaretools/hetest/hetest.html>, and we welcome feedback about our software at `hetest@ll.mit.edu`.

Acknowledgements. The authors would like to thank the following people:

- Tim Meunier, for writing the parsers for the test harness output that transfer the test data into the SQLite database.
- Oliver Dain, Nick Hwang and Ben Price, for their help with code reviews and general guidance throughout the software engineering process.
- Mike Depot and John O’Connor, for their IT support during the tests.

References

1. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D.J.: Private database queries using somewhat homomorphic encryption. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 102–118. Springer, Heidelberg (2013)
2. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 309–325. ACM, New York (2012). <http://doi.acm.org/10.1145/2090236.2090262>
3. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2011)
4. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Rošu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013)
5. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
6. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
7. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009). <https://www.crypto.stanford.edu/craig>
8. Gentry, C., Halevi, S.: Implementing gentry’s fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
9. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
10. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
11. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 1–16. Springer, Heidelberg (2012)
12. Halevi, S., Shoup, V.: HElib. <https://github.com/shaih/HElib>. Accessed 23 September 2014
13. Halevi, S., Shoup, V.: Algorithms in HElib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014)
14. IARPA: Broad agency announcement IARPA-BAA-11-01: Security and privacy assurance research (SPAR) program, February 2011. <https://www.fbo.gov/notices/c55e38dbde30cb668f687897d8f01e69>

15. MIT Lincoln Laboratory: Hetest, February 2011. <https://www.ll.mit.edu/mission/cybersec/softwaretools/hetest/hetest.html>
16. Open Source Initiative: The BSD 2-clause license. <http://opensource.org/licenses/BSD-2-Clause>. Accessed 23 September 2014
17. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
18. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: (SOSP 2011) ACM Symposium on Operating Systems Principles (2011)
19. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science, January 1979. <http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-212.pdf>
20. Raykova, M., Cui, A., Vo, B., Liu, B., Malkin, T., Bellovin, S.M., Stolfo, S.J.: Usable, secure, private search. IEEE Secur. Priv. **10**(5), 53–60 (2012)
21. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Found. Secur. Comput. **4**(11), 169–180 (1978)
22. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
23. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
24. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2011)
25. Varia, M., Price, B., Hwang, N., Cunningham, R., Hamlin, A., Herzog, J., Poland, J., Reschly, M., Yakoubov, S.: Automated assessment of secure search systems. Oper. Syst. Rev. (OSR) **49**(1), 22–30 (2015). Special Issue on Repeatability and Sharing of Experimental Artifacts
26. Yang, Y.: Evaluation of Somewhat Homomorphic Encryption Schemes. Master's thesis, Massachusetts Institute of Technology (2013)

Users' Privacy Concerns About Wearables

Impact of Form Factor, Sensors and Type of Data Collected

Vivian Genaro Motti^(✉) and Kelly Caine

School of Computing, Clemson University, Clemson, USA
`{vgenaro, caine}@clemson.edu`

Abstract. Wearables have become popular in several application domains, including healthcare, entertainment and security. Their pervasiveness, small size and autonomy, enlarges the potential of these devices to be employed in different activities and scenarios. Wearable devices collect data ubiquitously and continuously, about the individual user and also her surroundings, which can pose many privacy challenges that neither users nor stakeholders are ready to deal with. Before designing effective solutions for developing privacy-enhanced wearables, we need first to identify and understand *what* are the potential privacy concerns that users have and *how* they are perceived. To contribute to that purpose, in this paper we present findings from a qualitative content analysis of online comments regarding privacy concerns of wearable device users. We also discuss how form factors, sensors employed, and the type of data collected impact the users' perception of privacy. Our results show that users have different levels and types of privacy concerns depending on the type of wearable they use. By better understanding the users' perspectives about wearable privacy, we aim at helping designers and researchers to develop effective solutions to create privacy-enhanced wearables.

Keywords: Privacy · Wearable computing · Wearable devices · Form factors · Privacy concerns · User studies · Human factors

1 Introduction

The significant advances in technology in the past decades, characterized by the miniaturization of components, more efficient power sources, alternative network solutions and novel sensors, boosted the development of wearable devices. As a consequence, a variety of form factors have been created, enabling wearable devices to be applied for multiple different purposes. Despite the large potential and known benefits of wearable devices, their spread usage entails several privacy concerns. Wearables, by continuously collecting, transmitting and storing data, handle information that are often considered as personal, private, sensitive or confidential. This information can be publicly available or posted in social media, where it is shared with a network of friends of the individual user or even with unknown or untrusted parties. While the data collection and sharing brings many benefits for end users, it also brings novel privacy challenges for stakeholders involved in the creation of wearable devices and applications. Wearables enable the surveillance and sousveillance of individuals,

their behaviors and surroundings as well, which can lead to severe privacy implications, threats and risks. These issues affect not only the individual user but also the society and organizations involved, for instance when the data collected are misused. Due to the novelty of the wearable field such implications are not yet fully understood.

The continuous use of wearables involves a variety of privacy concerns, however because the usage of these devices is relatively recent, users are not aware of the potential privacy implications of continuous data collection, storage and online sharing. To better understand how users actually perceive wearable privacy and to identify what are their main concerns nowadays, we collected commentaries from users (end users and prospective users) from online sources (such as IT forums, websites, discussion lists and social medias) about several wearable devices (either commercially available or to be soon launched in the market) including head-mounted and wrist-mounted devices. With the analysis of the users' comments extracted from a set of online sources, we identified different concerns about wearable privacy, and we analyzed how they are related to specific form factors, sensors employed and data collected.

The main contributions of this paper consist in identifying: (i) what are the users' concerns for wearable privacy; (ii) how form factors, data collected and sensors employed impact these privacy concerns (regarding their levels and nature); and (iii) what concerns are specific to wearable devices, sensors and applications.

This paper is organized as follows: Sect. 2 motivates and contextualizes this research by presenting related works and the scope in which this research is inserted; Sect. 3 describes the method of the research; Sect. 4 presents the results obtained; Sect. 5 discusses them and Sect. 6 concludes.

2 Related Work

Privacy concerns are not exclusive from the technological domain, being discussed since 1890 [1]. Despite being in discussion for a long time, privacy issues related to mobile technologies are relatively new, complex to study and still poorly understood [2]. Moreover, mobile and wearable devices continuously collect data, spreading the use of sensors, such as: cameras, GPS, and accelerometers, whose small size and invisibility adds novel challenges to ensure users' privacy.

Most of the previous works on user privacy has focused on mobile devices and their applications [3], social networks [4, 5], web applications [6], or other security concerns, as account hijacking [7]. Little is known about wearable privacy [8, 9] from a human-centered perspective. Existing solutions frame the privacy problems too narrowly and satisfactory general solutions remain elusive [8], besides having a fragmented landscape [5]. The nature of privacy concerns remains an open question, requiring a better understanding of privacy behaviors in technology [10].

The following sections summarize related research findings, presenting and discussing privacy concerns and human perspectives in ubiquitous, mobile and wearable computing.

2.1 Privacy in Ubiquitous Computing

Characterized by the integration of computational solutions into the physical environment, ubiquitous computing enables inanimate objects to acquire intelligence, by

sensing, processing and communicating data [11]. These data concern the individual user and also her surroundings, and can imply in privacy issues. Despite the existence and importance of these issues, users have a limited understanding about those. By centering potential solutions for privacy-enhanced technologies on the users' perspectives and concerns, stakeholders can aid users to better understand and control their privacy in these systems [12].

2.2 Privacy in Mobile Devices

Significant improvements in mobile computing in the past decades popularized the use of mobile devices, with smart phones and mobile apps playing nowadays a fundamental and intimate role in users' everyday life. Despite the continuous data collection and transmission with these devices and apps, previous research shows that users are not aware about *what* data are collected and *how* they are used [3]. Despite the importance of mobile privacy concerns, they still remain poorly understood [2].

2.3 Privacy in Wearable Devices

Similarly to ubiquitous and mobile computing, in wearable computing, privacy is one of the main challenges yet to be solved [13]. Not only because wearable computers are able to sense, process and store intimate information about the users, but also because wearables are able to do it continuously and discreetly [14]. Besides this, currently, users cannot fully understand the potential risks, threats and implications involved with data collection and tend to underestimate those. However the data collected often enable to infer private information, especially when combined with other data, which can result in significant risks to the users' privacy [15].

As previous research identified, privacy became a key concern to users [14], being for instance among the top five concerns that users consider important in the wearability of HMD (head-mounted devices) [16]. Despite its relevancy, wearable privacy is still an emergent topic and many questions remain open.

Previous works related to wearable privacy have focused on its different aspects, including: (i) users' behaviors with wearable cameras, to identify the factors that impact how sensitive a photo is [9] and the privacy concerns in pictures illustrating eating behaviors of users [17]; (ii) requirements for remote communication in fashion garments [18]; (iii) perceptions of anklet wearers, to identify location-based privacy concerns [8]; (iv) privacy for augmented reality systems [19]; and v) surveillance concerns of Google Glass users [20]. Although these works aid to understand how users perceive wearable privacy, they focus on specific wearable devices or applications.

2.4 Users' Perspectives on Privacy

Privacy behaviors across multiple technologies were identified and analyzed in [10], aiding to understand the users' perspectives and concerns and to propose and devise novel solutions to ensure users' privacy. Despite extensive user studies, this work

targets a general understanding of privacy concerns, regardless of the technology employed. User studies were also conducted by [21], to better understand users' privacy concerns. This work, although focused on e-commerce applications, suggests a significant gap between reported concerns and actual users' behaviors, reinforcing that users often sacrifice their privacy in exchange of benefits. For [22], considering current users' needs and their cognitive models is key to ensure privacy control. The users' understanding about privacy was also analyzed in [6], but mainly regarding their interaction with web sites.

Despite previous works focusing on ubiquitous, mobile and wearable computing, it is still not clear what are the users' concerns about wearable privacy and how these are related to specific devices, sensors and applications. However, without understanding what the privacy problems are, privacy cannot be addressed in a meaningful way [23].

3 Methods

Because we were interested in assessing the privacy perceptions of a broad range of people who already had interest or experience in using wearable devices, and we wanted to gather a geographically and demographically diverse sample, we chose to conduct an observational study of online comments posted by wearable users. To identify concerns, we extracted comments from a series of online sources (described below). First, we selected a set of devices and data sources, then we identified, selected and analyzed the users' concerns about privacy in wearable devices. Further details about the methodology of this work are described in the following sections.

3.1 IRB Approval

To ensure the protection of human subjects, before data collection started, the Clemson University Institutional Review Board (IRB) approved this study as exempt.

3.2 Data Selection, Extraction and Analysis

Due to the fact the head-mounted and wrist-mounted devices are the most popular form factors available and in use today, we considered both form factors for data collection. To minimize bias in the data collected we selected a set of 59 different online sources, including popular websites for discussion and reviews of technology and e-commerce pages for shopping, reviewing and recommending devices. The data collection process resulted in more than 2,000 commentaries extracted for analysis. This process was conducted in April and May 2014 and consisted in visiting the online sources previously selected, searching for the users' comments regarding specific devices, and manually extracting the contents of interest to compose a report.

For the analysis, we filtered and selected the commentaries related to privacy concerns. For that a member of our research team read and analyzed each comment, to identify and annotate those related to privacy. Then, the annotated comments were analyzed again to identify the nature of the privacy concern regarding its motivations

and rationale behind it. For example, a user who fears the consequences of his/her location being posted online in a live feed through social media apps concerns the *Implications of Location Disclosure*). In order to identify the relationships between the privacy concerns and respective data collected and sensors, we analyzed the nature of the concerns identified and assessed whether they were specific to a given form factor and/or application or generic to mobile devices. The results of this analysis are graphically presented in a Venn diagram (Fig. 1).

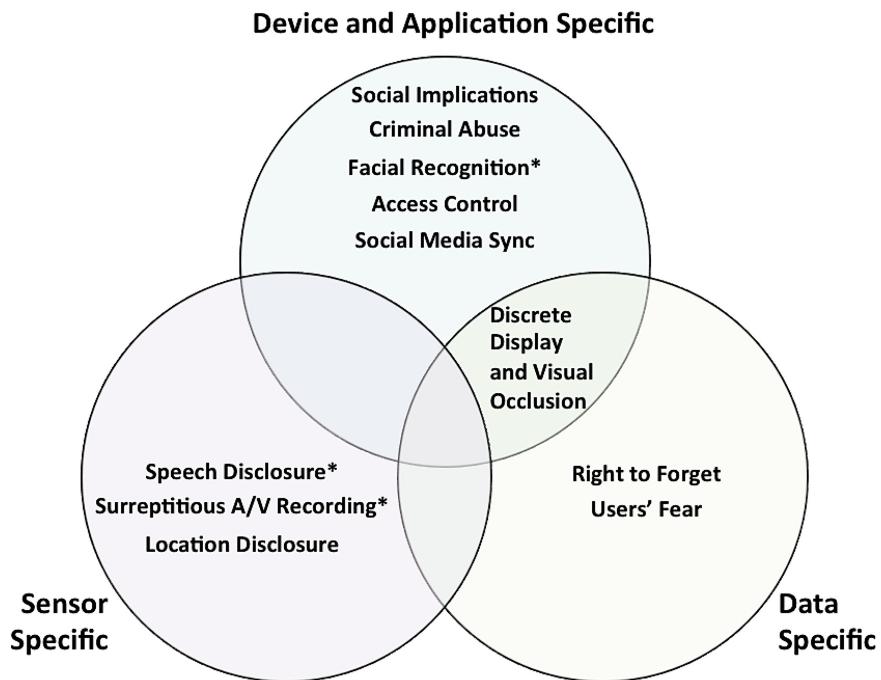


Fig. 1. Privacy concerns per form factor and according to their influencing factors: device, application, sensor or data. Concerns marked with a '*' were identified particularly for head-mounted devices.

3.3 Devices, Online Data Sources and Figures

The users' commentaries collected were generated at latest in May 2014. The date when a commentary was posted was not always available in the web sources selected, however users start commenting about a new device usually when a vendor announce it, launch it for sale or when a new release is made available. The comments collected were related to six wrist-mounted devices with different purposes, including:

- 27 privacy comments about six armbands and smart watches: Sony SWR10 (Core) Smartwear and Thalmic Labs Myo, Basis, Qualcomm Toq, LG Lifeband Touch, Razer Nabu.

The users' commentaries about the 32 head-mounted devices analyzed, included:

- 11 privacy comments about 19 earpieces, headbands and headphones: Looxcie, LG Lifeband Earphones, Intel Smart Earbuds, Microsoft Septimu, Avegant glyph, the Immersion headset, The Vigo, iRiver On, The Voyager Legend, NeuronOn, Recon's Snow 2, The Cynaps Enhance, iWinks Aurora Dreamband, Life Beam's Sports Headband, Emotiv Insight, Axio EEG Headband, Interaxon Muse, Muzik, Neurowear Zen tune Headphones;
- 34 privacy comments about 13 glasses: EmoPulse Nano Glass, Epson Moverio BT-200, Google Glass, Google Smart Contact Lenses, ICIS, K-Glass, Lesser See Tru, Meta Pro, Oculus Rift, Olympus MEG4.0, Second Sight's Argus II, Sony HMZ-T1, The Atheer One, Vuzix Smart Sun Glasses.

The 59 online sources used for data collection included 15 forums, 34 technical websites, 6 e-commerce websites, and 4 social medias, e.g.: Amazon, Ars Technica, BestBuy, CNET, ComputerWorld, DigitalTrends, ExpertReviews, Engadget, eWeek, Geek, GizMag, Gizmodo, Overstock, PCAdvisor, PCMag, PCPro, PCWorld, PhoneArena, Popular Science, Mashable, MIT Technology Review, Reddit, Slate, TechCrunch, TechRadar, TheInquirer, TheNextWeb, TheVerge, T3, TrustedReviews, Wearable Computing Review, Wearable Technologies, Wired, ZDNet. With a diversity of sources, we aimed at more representativeness in the data collected and minimizing the potential bias of commentaries that were not posted by actual users. The main differences among the comments consisted in: more extensive, detailed and formal comments produced by reviewers (posted in IT forums), and shorter, more informal and objective comments produced by end users and posted in e-commerce websites. By gathering comments from heterogeneous sources, we ensure a diversity of user profiles, and still focus on the study goals, covering a set of specific wearable devices and privacy concerns of users for different wearable applications.

The analysis of online reviews has some drawbacks, for instance, little is known about the profile of the user who posted a comment and we cannot ensure whether the comment was in fact generated by an individual user or by a bot, a spammer or even a competitor company, which can introduce bias in the study. In our analysis, to minimize this risk, we selected heterogeneous online sources (59 websites with high popularity) and an extensive list of comments ($n > 2,000$). Despite these drawbacks, as previous research indicate [24–27], the analysis of online reviews has several benefits as well, for instance: (i) users are placed in a *wild* study, i.e. not constrained by laboratory settings, (ii) the commentaries are self-reports of the users' opinions, without a standard format or pre-defined set of questions, and (iii) a large sample of reviews can be analyzed covering heterogeneous user profiles.

4 Identifying User Privacy Concerns for Wearable Technologies

The analysis of the online comments, revealed 13 users' concerns about wearable privacy. These concerns are closely related to the type of data each device collects, stores, processes and shares. Embedded sensors, such as cameras and microphones, capture data about the individual user or people nearby, often without their awareness

or consent. These data are oftentimes personal, confidential, and sensitive, which poses privacy challenges, for instance regarding surveillance. Other sensors, such as heart rate monitors, glucometers and activity trackers, are often considered by users as involving fewer privacy concerns.

By analyzing the users' commentaries, 13 privacy concerns emerged, six for wrist-mounted devices and seven for head-mounted devices. These concerns are presented in the following sections, ordered by form factor and the activity they are related to, respectively: data collection, data processing and data sharing (according to the three first groups of activities defined in the Solove's privacy taxonomy [23]).

4.1 Privacy Concerns for Wrist-Mounted Devices

Wrist-mounted devices collect data whose nature is less sensitive than head-mounted devices, at least in a first sight. Some HMDs are able to capture audio, image and videos, whose privacy implications tend to be more critical or at least apparent for users. WMDs, on the other hand, often include activity trackers and sense the user location, which is considered by users as less privacy-critical data. Actually, from our analysis, the GPS sensor is pointed as the most critical privacy concern for users of WMDs, as their location is sensed and stored, and sometimes even shared online in real time through live feeds of social media applications. Besides this, the form factors of wrist-mounted devices are similar to conventional accessories worn in a daily basis, such as watches and bracelets, so they fit seamlessly in conventional outfits of users, raising less attention or suspicion from other people. Among the six privacy concerns identified for WMDs and presented below, the two first ones are related to data collection and the other four refer to data sharing.

4.1.1 General Social Implications: Unawareness

An activity tracker that synchronizes data (e.g., location and photos) and relates it to the network of friends of an authenticated user, can also impact the privacy of other people (e.g., individuals belonging to the social network contacts of a given user):

'it does not just record your activities, but also activities of people around you, it can also connect to other devices'

The people belonging to the social network of a user are not necessarily aware of and compliant with the data collected, stored, published or shared.

4.1.2 Right to Forget

When data are continuously collected, stored, published and shared, they can include information that users do want to recall later, but also events and facts that users were not willing to capture or to be reminded of later on:

'it gives a record of everything you've done, day in and day out, possibly even some things you don't want to be reminded of'

4.1.3 Implications of Location Disclosure

The users' comments analyzed revealed that users were afraid that their location when tracked could be disclosed to malicious parties and criminals, such as thieves and

stalkers. These malicious parties could then misuse the user location, for instance to better plan a crime or other harmful actions:

'It [wearable device] just knows when to take pictures of the epic moments, know if you're riding in your car so your friends and stalkers know where you are at all times of the day, know when you go to sleep, riding a car, or climbing a mountain'

4.1.4 Discrete Display of Confidential Information: Non-Disclosure

Wrist-worn devices, such as smart watches, often use a screen to display notifications. These notifications can include sensitive or confidential information, which is also accessible to people located close to the end user. Being able to hide this information from co-located individuals is considered good for some users:

'the second screen will act as sort of a privacy screen, keeping folks from reading your texts by glancing at your wrist'

4.1.5 Lack of Access Control

Users who are aware about data storage in the cloud, fear that organizations or even the government will use their personal data without their awareness or consent, for instance for abusive or malicious purposes:

'[wearable devices are] the NSA's new best friend'

4.1.6 Users' Fears: Surveillance and Sousveillance

While most wearable device users acknowledge the many benefits of collecting and tracking their personal information, they fear the continuous surveillance and sousveillance and potential implications that this can bring them in the future:

*'I'm not sure if I should be totally excited or **totally frightened** about this Sony band logging my every move. I can't help but think it could be good ole big brother in disguise'*

4.2 Privacy Concerns for Head-Mounted Devices

Head-mounted devices that focus on augmented and virtual reality and gaming experiences did not raise as many privacy concerns for users (e.g., Oculus Rift and Sony HMZ-T1), because less sensitive data are collected, and also because the device does not store or share information, keeping it protected from social media and other online applications with networks of online users. On the other hand, head-worn computers, such as Google Glass, which are equipped with cameras and microphones, are often synchronized with a smart phone, allowing users to connect to social media applications. This results in several privacy concerns, as indicated our analysis of the users' commentaries. The next sections detail specific users' concerns with HMD. Among the seven users' concerns, the four first are related to data collection, one to data processing, and the last two refer to data sharing.

4.2.1 Speech Disclosure

Using speech recognition enables users to have a hands-free interaction, however when users are not alone and need to handle confidential information, audio as a unique input modality poses serious privacy concerns:

'though you can't mind people overhearing what you are saying'

4.2.2 Surveillance, Sousveillance and Criminal Abuse

By capturing data without any consent or awareness, users reported that they were concerned about a potential for criminal abuse:

'There are a lot of concerns about privacy invasion, spying and situations where people are more concerned with recording an event than actually engaging with it'

4.2.3 Surreptitious Audio and Video Recording: Unawareness

Although smart phones and mobile computers such as tablet PCs also include cameras and microphones, a HMD allows users to start recording content discreetly:

'the video camera that is even easier to use than a smartphone's ... privacy issues are indeed huge with that'

'Placing a tiny wearable device on someone's eye could potentially be a lot more discreet, though some privacy advocates might see that as a downside'

'I do believe there is a difference between snapping pictures with something which is obviously a camera, and recording video surreptitiously. Social norms already frown on making surreptitious audio recordings (though it isn't illegal, it is done only infrequently and with an air of "secret agency" about it); video is much more of an intrusion.'

'the more subtle and high tech augmented vision gets, the more dangerous it gets as well. Basically, we're teetering on a slippery slope here unless we find a solution for the privacy/harassment concern that is growing'

4.2.4 Surveillance, Sousveillance and Social Implications: Unawareness

The fact that the device captures information from the users' surrounding extends the privacy issues to the social environment, as people nearby are often unaware or not compliant with the data collection:

'There's also another challenge that affects not only those who wear Glass, but everyone else around: privacy'

Users may not feel comfortable to wear a device with a camera on their heads, at least nowadays and especially in environments in which this is not a common practice:

'The privacy concerns may very well be overblown, but I think it'll take a while for people to get comfortable with the idea of others walking around with camera-equipped devices strapped to their faces'

4.2.5 Facial Recognition: Identifiability

Users acknowledge the benefits of facial recognition to augment their memory, however, they are also aware that privacy concerns will likely emerge in the near future, as previously pointed out by [28]:

'...totally needs a camera. I want to be able to look at people and it have them tell me their names, limit it to my personal database of contacts if you must, but I'm terrible with names, if it wants to give me an immersive world experience, then it needs to be able to see what I see regardless of privacy worries.'

'Privacy officials understand that Google won't include facial recognition in Glass for now, but raised concerns about Google's future facial recognition plans'

'...image analysis. This of course raises all sorts of new privacy concerns with things like identifying people through facial recognition associated with Facebook pictures...'

4.2.6 Automatic Synchronization with Social Media: Linkability

Some users do not like the idea of their devices to immediately synchronize with social media applications and share their data without being able to control it:

'Why in the HELL would I ever want to tweet or facebook from a pair of headphones. Isn't there enough horror in the world without these in it?'

'Oh, how nice! Another unsubscribe factor to add to my unsubscribe rule list. Tweets from headphones? Unsubscribed!'

'Can't wait for the trend when not having a Facebook integration is a big thing...'

4.2.7 Visual Occlusion: Non-Disclosure

HMDs that cover the field of view of the user, e.g., Oculus Rift and Sony HMz-T1, allow users to interact privately because their vision is occluded:

'Not as a primary display but for those times when I really need some privacy'

'watch what they want in the privacy of their own rooms.'

'covered design will enable complete privacy for the viewer.'

'What I want is a head-mounted replacement for my laptop screen. So I don't have to have its size, weight, fragility, power consumption, and lack of privacy when I'm traveling'

'provide you some privacy for your augmented-reality browsing.'

4.3 Privacy Concerns Across Form Factors

The analysis of the users' comments collected resulted in 13 privacy concerns for wearable devices, some of them existing regardless of the form factor. By analyzing these concerns we noticed that some concerns (4) are device-specific, others (3) are sensor-specific, few (2) depend on the data collected, or (2) are both device/application- and data-specific:

- **Device-specific Privacy Concerns:** social implications (in general, devices that collect data that do not belong solely to an individual user, impact social aspects of privacy), criminal abuse (collecting personal data can facilitate criminal abuse), facial recognition can take place if the appropriate algorithms are available in the device, social media synchronization are not necessarily a user wish for wearables;
- **Sensor-specific Privacy Concerns:** location disclosure is associated with GPS usage, speech disclosure depends on the ability of using audio as input modality (HMD with a microphone), and surreptitious audio and video recording (HMD with cameras) depend on how invisible the sensors are embedded in a device, as data can be captured without it being noticeable;
- **Data-specific Privacy Concerns:** right to forget (all data that are collected without the consent, awareness or users' will should be able to be deleted after collection), users fear that certain data types when combined could have critical implications;

- **Device/Application and Data-specific Privacy Concerns:** discrete display and visual occlusion depends on devices with a screen available which should enable users to decide *what, when* and even *if* information can be displayed.

Most of the users' concerns, although identified in the analysis of one specific form factor, can also relate to different devices, depending usually on the availability of a specific sensor, feature or application. The location disclosure for instance depends mainly on a GPS to track users' location, which is usually embedded in a wrist-mounted device, but can be also found in an anklet or a helmet. Besides the GPS, other sensors or data sources can also be used to track the users' location. Figure 1 illustrates how the privacy concerns identified can be placed regarding their influencing factors.

5 Discussion

From the analysis of the users' concerns in wearable privacy we note that several factors affect the privacy concerns among users. These include: the nature of the data collected, their respective levels of confidentiality and sensitiveness, ability to share and disclose the information, and also potential implications (social, criminal, etc.).

The findings indicate that privacy concerns are not necessarily unique to one specific device or form factor, but are intimately related to the sensors embedded in the device and the respective data collected. We found that devices that include cameras and microphones resulted in more and more extreme privacy concerns, followed by devices with GPS and displays. Activity trackers that monitor heart rate, steps, and pulse for instance, are usually seen as inoffensive to the users' privacy, however it is likely that users are not aware of how such data could be misused by third-parties or potential privacy implications when the data are collected in a long term or associated with complementary information.

We also note a significant overlap between the users' concerns about mobile privacy and wearable privacy, mostly because the tasks that users can perform with wearables are also possible with alternative devices, which were previously used in a large scale (including cameras, pedometers, and tablet PCs). However from the analysis of the users' comments we do notice that specific characteristics of wearables strengthen these concerns. For example, while cameras and microphones were already employed in mobile devices, wearables make it easier to record data without other people noticing, so their lack of awareness, compliance and consent becomes more critical for privacy in the wearable context. Similarly, users have privacy concerns about location information, primarily because wrist-mounted devices are able to track their position and immediately publish it online in social media applications to a network of contacts. Users worry that this group can include malicious users and people that the individual user does not know or trust.

5.1 Limitations

An extensive list of 38 devices has been covered in the analysis of online comments, however, because the landscape of wearable devices is shifting very rapidly, obviously this list did not include every wearable device possible. For example, our analysis of

wrist-mounted devices included six devices, mainly armbands and smart watches. In future work, to complement our research findings, we plan to analyze fitness trackers as well, as we hypothesize that this specific type of WMD may pose more privacy concerns than smart watches and armbands currently do.

Although this work focuses mainly on head and wrist-mounted devices, we believe that chest and back-mounted devices, such as the Polar band for heart rate monitoring and Lumo back band for posture tracking could also raise privacy concerns. To observe this, in future work, we plan to verify potential privacy implications that such devices could involve, and identify potential users' concerns.

Collecting and analyzing online data is a relatively new research method, and despite enabling the analysis of large amounts of contents, it involves two main limitations: first, no well-established and validated protocol is available regarding data collection and analysis, so the method employed in this research is both exploratory and empirical. Second, little is known about the users' profiles, as all data collected are anonymous. However, we can assume that users who post online comments access frequently the web (forums, IT websites) and are interested in technology (to follow new trends and news in the domain). Despite being a niche of users, which limits the generalization of the research findings, they also correspond to actual or potential users interested in wearable technologies.

6 Conclusion

The analysis of the users' comments shows that the privacy concerns about wearables are similar, but in some cases more specific than privacy concerns about mobile devices. It also shows that users are aware about potential privacy implications, but mainly during data collection and sharing. The privacy concerns of users are related to the ability of the wearable device to sense, collect, and store data, which are often private, personal, confidential or sensitive, and then share these data with unknown or untrusted parties.

Users' concerns about wearable privacy cover different aspects of the user interaction with a wearable, including: disclosure of sensitive information, subtle data collection (of audio and video), public posts in social media apps (sharing), and lack of control and awareness regarding who has access to the data collected.

Although the level of privacy concerns of users is similar to that of mobile devices, the nature of their concerns is critical, showing that because users are somewhat unaware about potential privacy implications, vendors should alert them about possible problems, enabling them to apply a fine-grained control about *what* is collected, *when*, and *how*, and also *how* data are shared (*who* has access).

While there is a long way to go to build wearable devices and applications that are actually privacy-enhanced, this work brings insight in clarifying the users' concerns about wearable privacy, aiding to devise better solutions in the future.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. 1314342. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Warren, S.D., Brandeis, L.D.: Right to privacy. *Harv. Law Rev.* **4**, 193–220 (1890)
2. Mancini, C., Thomas, K., Rogers, Y., Price, B.A., Jedrzejczyk, L., Bandara, A.K., Joinson, A.N., Nuseibeh, B.: From spaces to places: emerging contexts in mobile privacy. In: Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp 2009), pp. 1–10. ACM, New York (2009). doi:[10.1145/1620545.1620547](https://doi.acm.org/10.1145/1620545.1620547), <http://doi.acm.org/10.1145/1620545.1620547>
3. Shklovski, I., Mainwaring, S.D., Skúladóttir, H.H., Borghorsson, H.: Leakiness and creepiness in app space: perceptions of privacy and mobile app use. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI 2014), pp. 2647–2656. ACM, New York (2014). doi:[10.1145/2556288.2557421](https://doi.acm.org/10.1145/2556288.2557421), <http://doi.acm.org/10.1145/2556288.2557421>
4. Ur, B., Wang, Y.: A cross-cultural framework for protecting user privacy in online social media. In: Proceedings of the 22nd International Conference on World Wide Web Companion (WWW 2013 Companion), pp. 755–762. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva (2013)
5. Gürses, S., Diaz, C.: Two tales of privacy in online social networks. *IEEE Secur. Priv.* **11**(3), 29–37 (2013). doi:[10.1109/MSP.2013.47](https://doi.org/10.1109/MSP.2013.47), <http://dx.doi.org/10.1109/MSP.2013.47>
6. Reidenberg, J.R. et al.: Disagreeable Privacy Policies: Mismatches between Meaning and Users' Understanding, 15 August 2014. Fordham Law Legal Studies. At: <http://ssrn.com/abstract=2418297>
7. Shay, R., Ion, I., Reeder, R.W., Consolvo, S.: My religious aunt asked why I was trying to sell her viagra: experiences with account hijacking. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2014), pp. 2657–2666. ACM, New York (2014). doi:[10.1145/2556288.2557330](https://doi.acm.org/10.1145/2556288.2557330), <http://doi.acm.org/10.1145/2556288.2557330>
8. Troshynski, E., Lee, C., Dourish, P.: Accountabilities of presence: reframing location-based systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2008), pp. 487–496. ACM, New York (2008). doi:[10.1145/1357054.1357133](https://doi.acm.org/10.1145/1357054.1357133), <http://doi.acm.org/10.1145/1357054.1357133>
9. Hoyle, R., Templeman, R., Armes, S., Anthony, D., Crandall, D., Kapadia, A.: Privacy behaviors of lifeloggers using wearable cameras. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2014), pp. 571–582. ACM, New York (2014). doi:[10.1145/2632048.2632079](https://doi.acm.org/10.1145/2632048.2632079), <http://doi.acm.org/10.1145/2632048.2632079>
10. Caine, K.: Exploring everyday privacy behaviors and disclosures. Ph.D. thesis. Georgia Institute of Technology (2009)
11. Schaub, F.M.: Dynamic privacy adaptation in ubiquitous computing. Ph.D. thesis. Universität Ulm (2014)
12. Könings, B., Schaub, F., Weber, M.: Who, how, and why? Enhancing privacy awareness in ubiquitous computing. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 364–367 (2013). doi:[10.1109/PerComW.2013.6529517](https://doi.org/10.1109/PerComW.2013.6529517)
13. Starner, T.: The challenges of wearable computing: part 1. In: *IEEE Micro* **21**(4), 44–52 (2001). doi:[10.1109/40.946681](https://doi.org/10.1109/40.946681)
14. Starner, T.: The challenges of wearable computing: part 2. In: *IEEE Micro* **21**(4), 54–67 (2001). doi:[10.1109/40.946683](https://doi.org/10.1109/40.946683)

15. Raji, A., Ghosh, A., Kumar, S., Srivastava, M.: Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2011), pp. 11–20. ACM, New York (2011). doi:[10.1145/1978942.1978945](https://doi.acm.org/10.1145/1978942.1978945), <http://doi.acm.org/10.1145/1978942.1978945>
16. Motti, V.G., Caine, K.: Understanding the wearability of head-mounted devices from a human-centered perspective. In: Proceedings of the 2014 ACM International Symposium on Wearable Computers (ISWC 2014), pp. 83–86. ACM, New York (2014). doi:[10.1145/2634317.2634340](https://doi.acm.org/10.1145/2634317.2634340), <http://doi.acm.org/10.1145/2634317.2634340>
17. Thomaz, E., Parnami, A., Bidwell, J., Essa, I., Abowd, G.D.: Technological approaches for addressing privacy concerns when recognizing eating behaviors with wearable cameras. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), pp. 739–748. ACM, New York (2013). doi:[10.1145/2493432.2493509](https://doi.acm.org/10.1145/2493432.2493509), <http://doi.acm.org/10.1145/2493432.2493509>
18. Jacob, C., Dumas, B.: Designing for intimacy: how fashion design can address privacy issues in wearable computing. In: Proceedings of the 2014 ACM International Symposium on Wearable Computers: Adjunct Program (ISWC 2014 Adjunct), pp. 185–192. ACM, New York (2014). doi:[10.1145/2641248.2641353](https://doi.acm.org/10.1145/2641248.2641353), <http://doi.acm.org/10.1145/2641248.2641353>
19. Roesner, F., Kohno, T., Molnar, D.: Security and privacy for augmented reality systems. Commun. ACM, **57**(4), 88–96 (2014). doi:[10.1145/2580723.2580730](https://doi.acm.org/10.1145/2580723.2580730), <http://doi.acm.org/10.1145/2580723.2580730>
20. Mcnaney, R., Vines, J., Roggen, D., Balaam, M., Zhang, P., Poliakov, I., Olivier, P.: Exploring the acceptability of google glass as an everyday assistive device for people with parkinson. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2014), pp. 2551–2554. ACM, New York (2014). doi:[10.1145/2556288.2557092](https://doi.acm.org/10.1145/2556288.2557092), <http://doi.acm.org/10.1145/2556288.2557092>
21. Berendt, B., Günther, O., Spiekermann, S.: Privacy in e-commerce: stated preferences vs. actual behavior. In: Commun. ACM **48**(4), 101–106 (2005). doi:[10.1145/1053291.1053295](https://doi.acm.org/10.1145/1053291.1053295)
22. Nguyen, D., Mynatt, E.: Privacy mirrors: understanding and shaping socio-technical ubiquitous computing systems. Georgia Institute of Technology GVU Technical Report (GIT-GVU-02-16) (2002)
23. Solove, D.J.: Understanding Privacy. Harvard University Press, May 2008. GWU Legal Studies Research Paper No. 420, GWU Law School Public Law Research Paper No. 420. SSRN: <http://ssrn.com/abstract=1127888> (2008)
24. Hedegaard, S., Simonsen, J.: Extracting usability and user experience information from online user reviews. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2013), pp. 2089–2098. ACM, New York (2013). doi:[10.1145/2470654.2481286](https://doi.acm.org/10.1145/2470654.2481286), <http://doi.acm.org/10.1145/2470654.2481286>
25. Jacob, C., Veerappa, V., Harrison, R.: What are you complaining about?: a study of online reviews of mobile applications. In: Proceedings of the 27th International BCS Human Computer Interaction Conference (BCS-HCI 2013). British Computer Society, Swinton, p. 6 (2013). Article 29
26. Fu, B., Lin, J., Li, L., Faloutsos, C.: Why people hate your app: making sense of user feedback in a mobile app store. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2013), pp. 1276–1284. ACM, New York (2013). doi:[10.1145/2487575.2488202](https://doi.acm.org/10.1145/2487575.2488202), <http://doi.acm.org/10.1145/2487575.2488202>
27. Khalid, H., Shihab, E., Nagappan, M., Hassan, A.: What do mobile app users complain about? a study on free iOS apps. In: IEEE Softw. 32(3), 70–77 (2015). doi:[10.1109/MS.2014.50](https://doi.org/10.1109/MS.2014.50)
28. Acquisti, A., Gross, R., Stutzman, F.: Faces of facebook: privacy in the age of augmented reality. In: BlackHat Webcast Series USA (2011)

On Vulnerabilities of the Security Association in the IEEE 802.15.6 Standard

Mohsen Toorani^(✉)

Department of Informatics, University of Bergen, Bergen, Norway
mohsen.toorani@uib.no

Abstract. Wireless Body Area Networks (WBAN) support a variety of real-time health monitoring and consumer electronics applications. The latest international standard for WBAN is the IEEE 802.15.6. The security association in this standard includes four elliptic curve-based key agreement protocols that are used for generating a master key. In this paper, we challenge the security of the IEEE 802.15.6 standard by showing vulnerabilities of those four protocols to several attacks. We perform a security analysis on the protocols, and show that they all have security problems, and are vulnerable to different attacks.

Keywords: Wearable security · Cryptographic protocols · Authenticated Key Exchange · Elliptic curves · Attacks

1 Introduction

Advances in wireless communication and embedded computing technologies, such as wearable and implantable biosensors, have enabled the design, development, and implementation of *Body Area Networks* (BAN) [1]. A BAN, also referred to as a *Wireless Body Area Network* (WBAN) or a *Body Sensor Network* (BSN), is a wireless network of wearable computing devices. BAN devices may be embedded inside the body (implants), may be mounted on the body (wearable technology), or may be accompanying devices that humans can carry in clothes pockets, by hand or in various bags. WBAN can be used for many applications such as military, ubiquitous health care, sport, and entertainment [1, 2]. WBANs have a huge potential to revolutionize the future of health care monitoring by diagnosing many life-threatening diseases, and providing real-time patient monitoring [2]. WBANs may interact with the Internet and other existing wireless technologies.

The latest standardization of WBANs is the IEEE 802.15.6 standard [3] which aims to provide an international standard for low power, short range, and extremely reliable wireless communication within the surrounding area of the human body, supporting a vast range of data rates for different applications.

The network topology consists of nodes and hubs. A node is an entity that contains a Medium Access Control (MAC) sublayer and a physical (PHY) layer, and optionally provides security services. A hub is an entity that possesses a node's

functionality, and coordinates the medium access and power management of the nodes. Nodes can be classified into different groups based on their functionality (personal devices, sensors, actuators), implementation (implant nodes, body surface nodes, external nodes) and role (coordinators, end nodes, relays) [2].

Although security is a high priority in most networks, little study has been done in this area for WBANs. As WBANS are resource-constrained in terms of power, memory, communication rate and computational capability, security solutions proposed for other networks may not be applicable to WBANs. Confidentiality, authentication, integrity, and freshness of data together with availability and secure management are the security requirements in WBAN [2]. A *security association* is a procedure in the IEEE 802.15.6 standard to identify a node and a hub to each other, to establish a new *Master Key* (*MK*) shared between them, or to activate an existing *MK* pre-shared between them. The security association in the IEEE 802.15.6 standard is based on four key agreement protocols that are presented in the standard [3].

Authenticated Key Exchange (AKE) and Password-Authenticated Key Exchange (PAKE) protocols aim to exchange a cryptographic session key between legitimate entities in an authenticated manner. Several security properties must be satisfied by AKE and PAKE protocols, and they should obviously withstand well-known attacks. Many protocols have been proposed in the literature, but some of them have been shown to have security problems [4–6]. It is desirable for AKE protocols to provide known-key security, forward secrecy, key control, and resilience to well-known attacks such as Key-Compromise Impersonation (KCI) and its variants, unknown key-share (UKS), replay, and Denning-Sacco attacks. PAKE protocols must also be resilient to dictionary attacks [7,8].

In this paper, we perform a security analysis on four key agreements protocols that are used in the security association process of the IEEE 802.15.6 standard [3]. We challenge the security of the IEEE 802.15.6 standard by showing vulnerabilities of those four protocols to several attacks. Excluding vulnerability of the first protocol to the impersonation attack which has been implied in the standard, no attack or security vulnerability has been reported in the standard or literature. All the protocols are available in the latest version of the IEEE 802.15.6 standard. The rest of this paper is organized as follows. We review the security structure of the IEEE 802.15.6 standard in Sect. 2, these key agreement protocols in Sect. 3, and report their security problems in Sect. 4.

2 Security Structure of the IEEE 802.15.6 Standard

The Security hierarchy of the IEEE 802.15.6 standard is depicted in Fig. 1. All nodes and hubs must choose three security levels: unsecured communication (level 0), authentication but no encryption (level 1), and authentication and encryption (level 2). During the security association process, a node and a hub need to jointly select a suitable security level. In unicast communication, a pre-shared or a new *MK* is activated. A *Pairwise Temporal Key* (*PTK*) is then generated that is used only once per session. In multicast communication, a

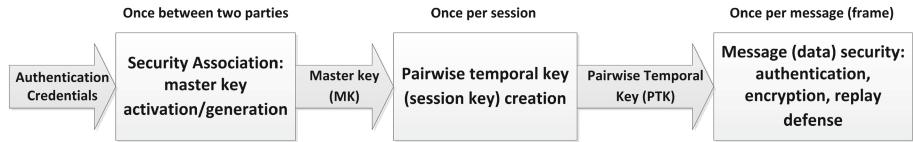


Fig. 1. Security hierarchy in IEEE 802.15.6 Standard [3]

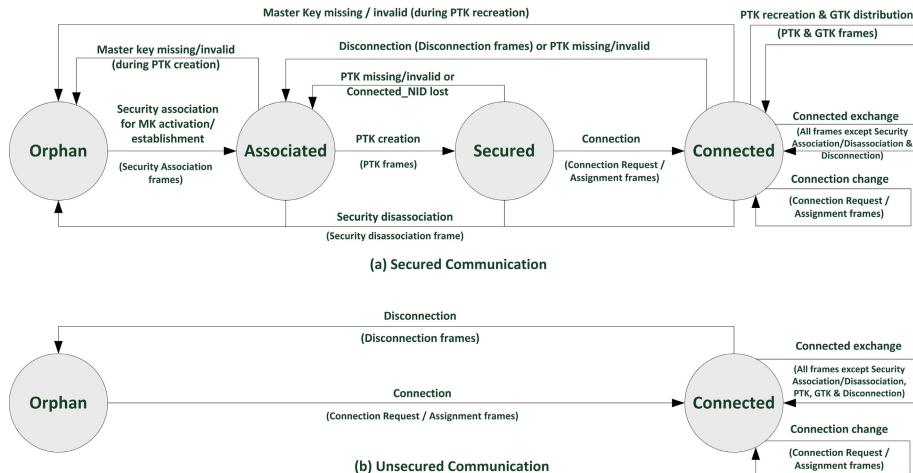


Fig. 2. MAC and security state diagrams in IEEE 802.15.6 Standard [3]

Group Temporal Key (GTK) is generated that is shared with the corresponding group [3]. All nodes and hubs in a WBAN have to go through certain stages at the MAC layer before data exchange. The security state diagrams of the IEEE 802.15.6 Standard for secured and unsecured communication are depicted in Fig. 2. In a secured communication, a node can be in one of following states [3]:

- *Orphan*: The initial state where the node does not have any relationship with the hub for secured communication. The node should activate a pre-shared *MK* or share a new *MK* with the hub. They cannot proceed to the *Associated* state if they fail to activate/establish a shared *MK*.
- *Associated*: The node holds a shared *MK* with the hub for their *PTK* creation. The node and hub are allowed to exchange *PTK* frames with each other to confirm the possession of a shared *MK*, create a *PTK* and transit to the *Secured* state. If the *MK* is invalid/missing during the *PTK* creation, they will move back to the *Orphan* state.
- *Secured*: The node holds a *PTK* with the hub. The node and hub can exchange security disassociation frames, connection assignment secure frames, connection request and control unsecured frames. The node can exchange Connection Request and Connection Assignment frames with the hub to form a connection, and transit to the *Connected* state.

- *Connected*: The node holds an assigned *Connected_NID*, a wakeup arrangement, and optionally one or more scheduled and unscheduled allocations with the hub for abbreviated node addressing, desired wakeup, and optionally scheduled and unscheduled access. The node and hub are not allowed to send any unsecured frame to each other, other than unsecured control frames if authentication of control type frames was not selected during the association.

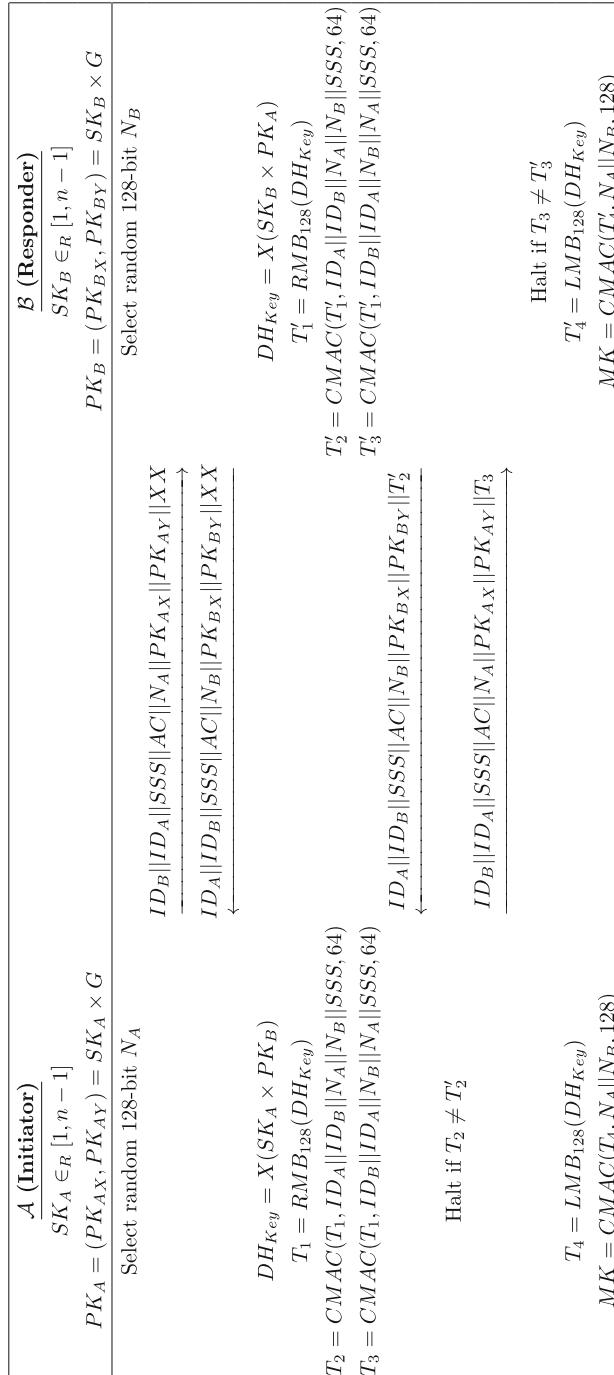
3 Key Agreement Protocols in the IEEE 802.15.6 Standard

The security association in the 802.15.6 standard involves a Master Key (MK) which is generated using one of four two-party key agreement protocols, proposed in the standard. Those four protocols, that will be referred to as protocols I-IV in this paper, are depicted in Figs. 3, 4, 5 and 6. The goal is to establish a new MK between a node and a hub. The node and hub are denoted by \mathcal{A} and \mathcal{B} , respectively. For simplicity, we have used simpler notations than those of the standard [3]. We have deleted *Immediate Acknowledgement* (I-Ack) messages that \mathcal{B} sends to \mathcal{A} , after receiving each frame from \mathcal{A} . I-Ack is kind of control type frames, and consists of current allocation slot number (8 bits) and current allocation slot offset (16 bits). We deleted I-Ack because they are sent in clear. Any information sent in clear, can be deleted from the security analysis. Protocols I-IV are similar, but vary in details and requirements. Protocol I is unauthenticated, and does not have any special requirement. Protocol II requires pre-shared and out-of-band transfer of a node's public key to the hub. Then, it is assumed that a hub obtains a node's public key via a separate protected channel, and a hub needs to save public keys of the nodes. Protocol III requires that a node and hub pre-share a password (PW). Protocol IV requires that \mathcal{A} and \mathcal{B} each has a display that shows a decimal number. It also requires that before accepting a new MK , human user(s) verify that both displays show the same number.

Protocols I-IV are based on elliptic curve public key cryptography. The domain parameters consist of an elliptic curve with Weierstrass equation of the form $y^2 = x^3 + ax + b$, defined over the finite field $GF(p)$ where p is a prime number. In order to make the elliptic curve non-singular, $a, b \in GF(p)$ should satisfy $4a^3 + 27b^2 \neq 0$. There are other conditions that should be satisfied in order to avoid known attacks on elliptic curve-based schemes [9]. The base point G in the elliptic curve is of order n , where $n \times G = O$ in which O denotes the point at infinity. The IEEE 802.15.6 standard suggests using the Curve P-256 in FIPS Pub 186-3. Values of a, b, p, n and G are public, and given in [3].

The private keys shall be 256-bit random integers, chosen independently from the set of integers $\{1, \dots, n - 1\}$. The private key of \mathcal{A} and \mathcal{B} is denoted by SK_A and SK_B , respectively. The corresponding public keys are generated as $PK_A = (PK_{AX}, PK_{AY}) = SK_A \times G$ and $PK_B = (PK_{BX}, PK_{BY}) = SK_B \times G$.

The IEEE 802.15.6 standard does not include having a digital certificate for public keys. Public keys are self-generated by involved parties, and are not

**Fig. 3.** Unauthenticated key agreement protocol (Protocol I).

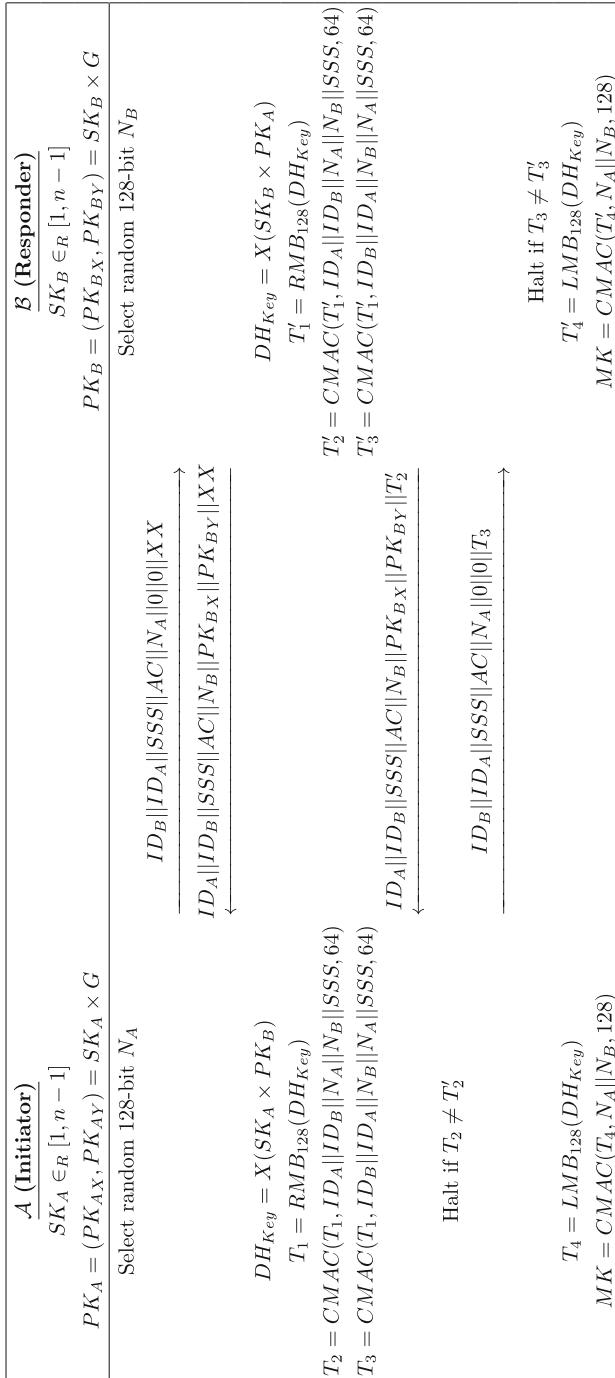
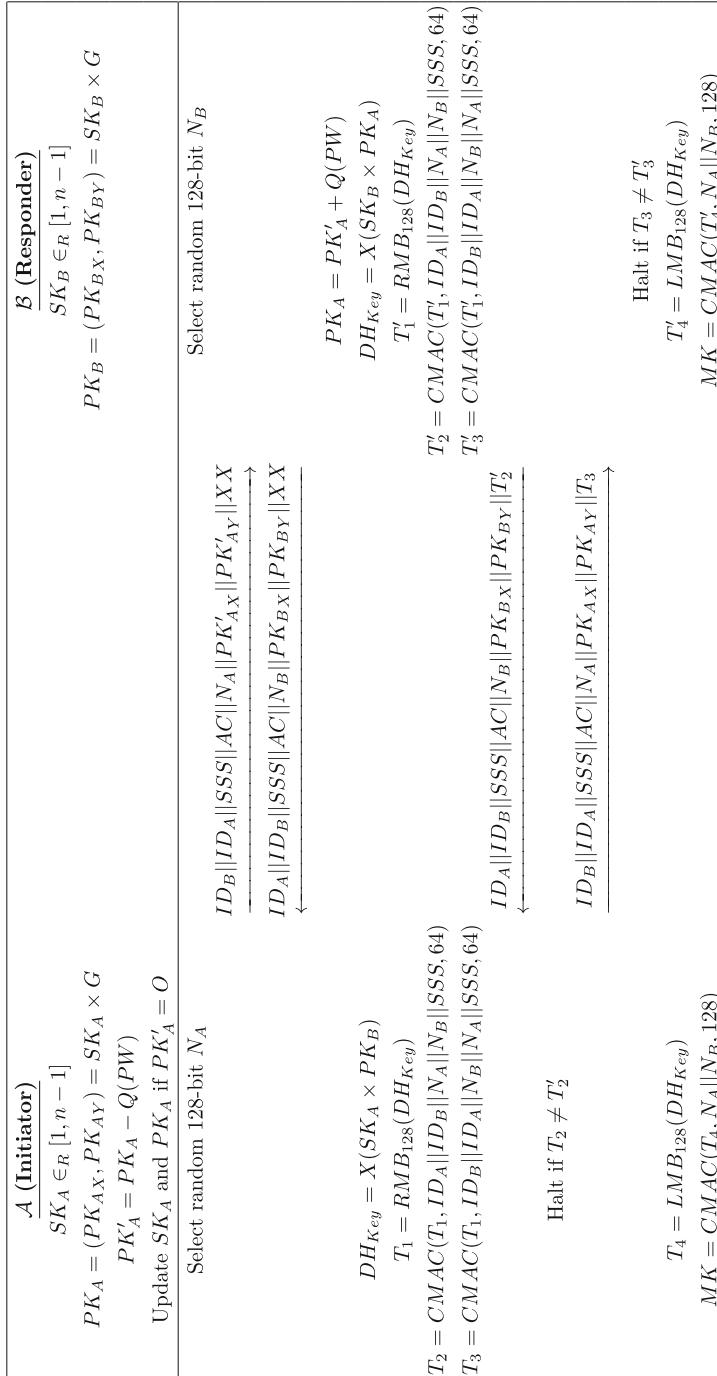


Fig. 4. Hidden public key transfer authenticated key agreement protocol (Protocol II).

**Fig. 5.** Password authenticated association procedure (Protocol III).

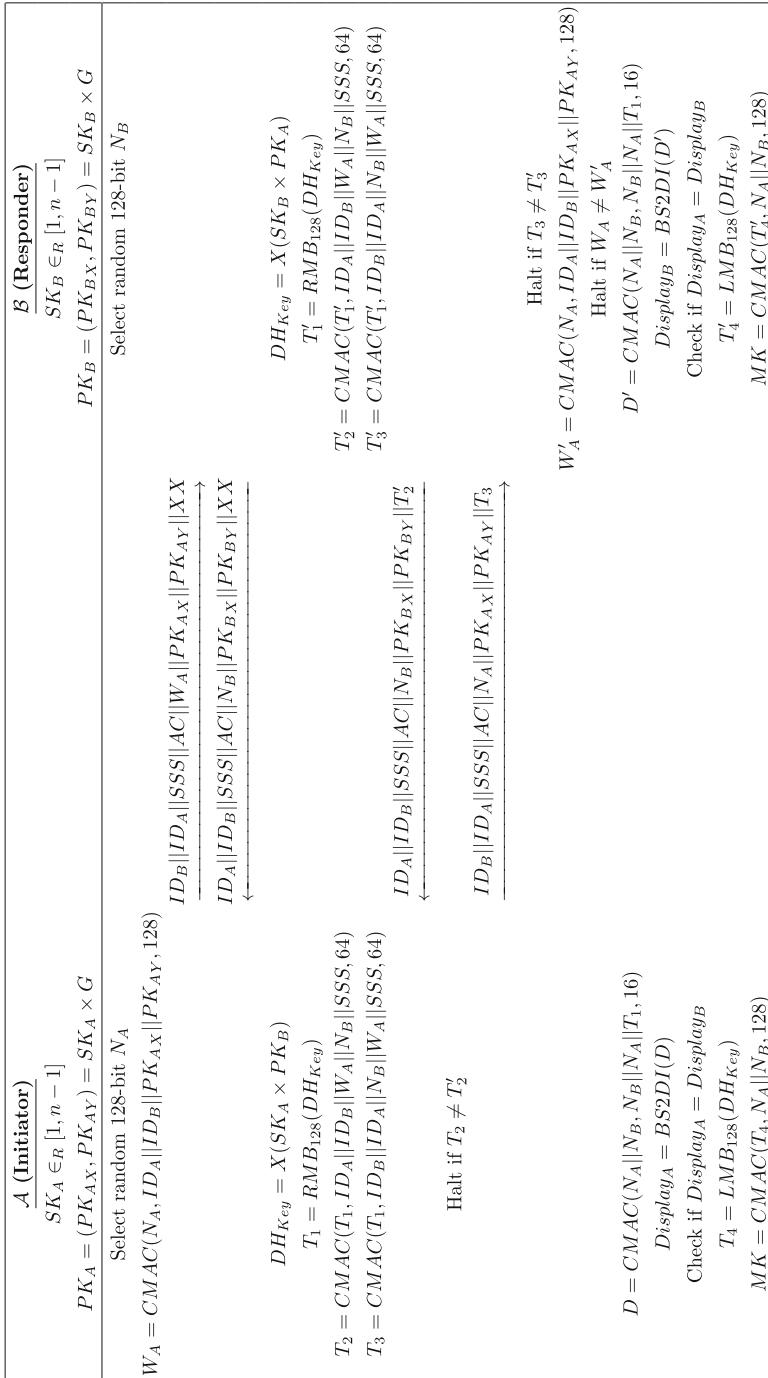


Fig. 6. Display authenticated association procedure (Protocol IV).

accompanied by digital certificates. It is because nodes are likely to be severely resource-constrained, and hence cannot store certificates or perform the certificate validation. The process of certificate validation consists of verifying the integrity and authenticity of the certificate by verifying the certificate authority's signature on the certificate, verifying that the certificate is not expired, and verifying that the certificate is not revoked [9].

The standard specifies that the node and the hub will abort execution of the protocols if the received public key, sent from the other party, is not a valid public key. A received public key $PK_i = (PK_{iX}, PK_{iY})$ shall be treated valid only if it is a non-infinity point (i.e. $PK_i \neq O$) on the defined elliptic curve, i.e. PK_{iX} and PK_{iY} satisfy the elliptic curve equation given above. This has been explained in protocol descriptions in the IEEE 802.15.6 standard, but they are absent in the corresponding figures of the standard. We do not show such verifications in Figs. 3, 4, 5 and 6 either. It is noteworthy that validation of elliptic curve public keys includes more steps than those mentioned in the standard. In addition to those conditions, one should check that PK_{iX} and PK_{iY} are properly represented elements in $GF(p)$, and that $n \times PK_i = O$. The last condition is implied by the other three conditions if the cofactor of the elliptic curve $h = 1$, which is the case for curves over prime finite fields [10].

In protocols I-IV, \mathcal{B} always sends his public key PK_B in clear. In Protocols I and IV, \mathcal{A} sends her public key in clear. In Protocols II, \mathcal{A} does not send her public key, as PK_A is pre-shared with \mathcal{B} . However, in protocol III, \mathcal{A} first sends a masked public key $PK'_A = PK_A - Q(PW)$ in which PW is a positive integer, converted from the pre-shared password between \mathcal{A} and \mathcal{B} . PW is converted according to the IEEE 1363-2000 standard from the UTE-16BE representation, specified in ISO/IEC 10646:2003, by treating the leftmost octet as the octet containing the *Most Significant Bits* (MSB). The $Q(.)$ function is a mapping which converts the integer PW to the point $Q(PW) = (Q_X, Q_Y)$ on the elliptic curve in which $Q_X = 2^{32}PW + M_X$ where M_X is the smallest nonnegative integer such that Q_X becomes the X-coordinate of a point on the elliptic curve. Q_Y is an even positive integer, and is the Y-coordinate of that point. In protocol III, \mathcal{A} shall choose a private key SK_A such that the X-coordinate of PK_A is not equal to the X-coordinate of $Q(PW)$.

$CMAC(K, M, L)$ represents the L -bit output of the *Cipher-based Message Authentication Code* (CMAC), applied under key K to message M . The standard suggests to use CMAC with the AES forward cipher function as specified in the NIST SP800-38B, and to use a 128-bit key as specified in FIPS197. $LMB_L(S)$ and $RMB_L(S)$ designates the L leftmost and the L rightmost bits of the bit string S , respectively. $X(P)$ denotes the X-coordinate of point P on the elliptic curve, i.e. $X(P) = X(P_X, P_Y) = P_X$. The sign \parallel denotes concatenation of bit strings that are converted according to the IEEE 1363-2000 standard. $BS2DI(BS)$ converts the bit string BS to a positive decimal integer for display. SSS is the *Security_Suite_Selector* (16 bits), AC is the *Association_Control* (16 bits), and XX is $X0000000000000000$. *Security_Suite_Selector* specifies type of cryptographic algorithms and protocols

that will be used during the protocol execution. It consists of the type of security association protocol (3 bits), i.e. binary representation of the protocol type according to our numbering I-IV, security level (2 bits), Control Frame Authentication (1 bit), cipher function (4 bits), and 6 bits reserved for future uses. *Association_Control* consists of Association Sequence Number (4 bits), Association Status (4 bits), and 8 bits reserved for future. *SSS* is fixed during a protocol execution, but *AC* will be different for each message. This is because *AC* includes the Association Sequence Number which is increased by one after each frame is sent during a protocol execution. Excluding I-Ack frames that are deleted from the protocols, there are four paths between \mathcal{A} and \mathcal{B} in all protocols.

4 Security Problems

In this section, we show that protocols I-IV are vulnerable to several attacks. All the protocols are vulnerable to the KCI attack, and they do not provide the forward secrecy. Furthermore, protocols I, III and IV are vulnerable to the impersonation attack. Protocol III is also vulnerable to an offline dictionary attack. Excluding vulnerability of protocol I to the impersonation attack which has been implied in the standard, no attack has been reported on the protocols, and they are available in the IEEE 802.15.6 standard.

The impersonation attack is feasible because public keys are self-generated by involved parties, and they are not accompanied by digital certificates due to resource constraints in the nodes. Although this is not recommended in the standard, if one can use certified public keys, or we can have a lightweight PKI [11,14], this can prevent the impersonation attacks. However, all the protocols will still be vulnerable to the KCI attack. The KCI attack is a variant of the impersonation attack, and has been considered in the eCK security model [12] for AKE protocols. Resilience to the KCI attack is an important security attribute for AKE protocols. If the private key of an entity \mathcal{A} is compromised, an adversary \mathcal{M} can impersonate \mathcal{A} in one-factor authentication protocols. However, such compromise should not enable \mathcal{M} to impersonate other honest entities in communication with \mathcal{A} . For the sake of brevity, we skip description of the KCI attack on protocols I, III and IV, because they are already vulnerable to the impersonation attack which is stronger than the KCI attack.

Forward secrecy is an important security attribute in AKE protocols. If an entity's private key has been compromised, it should not affect the security of session keys that have been established before the compromise. We have also the notion of *Perfect Forward Secrecy* (PFS) which is a bit stronger than the forward secrecy. PFS means that the established session keys should remain secure even after compromising the private keys of all the entities that are involved in the protocol. We have the concept of weak-PFS which only allows a passive attack after compromise of all involved private keys.

Protocols I-IV use elliptic curve cryptography. Then, it is crucial to have the public key validation. Upon receiving an ephemeral or static public key, the recipient entity must validate it. Otherwise, the protocol would be vulnerable

to further attacks. In description of protocols I-IV in the IEEE 802.15.6 standard, it has been mentioned that public keys should be validated. However, such validations are absent in corresponding figures in the standard. If one implements the protocols according to the standard's figures, and does not consider public key validations, further security vulnerabilities will arise. There will be extra scenarios for impersonation attacks on the protocols. Furthermore, all the protocols will be vulnerable to an invalid-curve attack [13] whereby an attacker can extract the private key of another entity. We do not consider those extra vulnerabilities, and strongly recommend to validate public keys.

In the rest of this paper, \mathcal{E} denotes the adversary in a passive attack, and \mathcal{M} denotes the adversary in an active attack. The order of protocols and attacks does not imply any preference or importance. The numbering is according to the standard, and will be included in the *SSS* during protocol executions.

4.1 Protocol I

Protocol I is an unauthenticated key exchange protocol. It is trivially vulnerable to an impersonation attack, but we consider it just for completeness of our security analysis. Such vulnerability has been implied in the standard only for this protocol, where the protocol is introduced as a protocol “without the benefit of keeping third parties from launching impersonation attacks” [3]. Protocol I does not provide the forward secrecy.

Impersonation Attack: Here is an impersonation attack on protocol I, in which \mathcal{M} impersonates \mathcal{A} :

- \mathcal{M} selects a private key SK_M , and generates the corresponding public key as $PK_M = (PK_{MX}, PK_{MY}) = SK_M \times G$. \mathcal{M} selects a 128-bit random number N_M , and sends $\{ID_B||ID_A||SSS||AC||N_M||PK_{MX}||PK_{MY}||XX\}$ to \mathcal{B} .
- \mathcal{B} selects a 128-bit random number N_B , and sends $\{ID_A||ID_B||SSS||AC||N_B||PK_{BX}||PK_{BY}||XX\}$ to \mathcal{M} .
- \mathcal{B} computes $DH_{Key} = X(SK_B \times PK_M)$, $T'_1 = RMB_{128}(DH_{Key})$, $T'_2 = CMAC(T'_1, ID_A||ID_B||N_M||N_B||SSS, 64)$, and $T'_3 = CMAC(T'_1, ID_B||ID_A||N_B||N_M||SSS, 64)$. \mathcal{B} sends $\{ID_A||ID_B||SSS||AC||N_B||PK_{BX}||PK_{BY}||T'_2\}$ to \mathcal{M} .
- \mathcal{M} computes $DH_{Key} = X(SK_M \times PK_B)$, $T_1 = RMB_{128}(DH_{Key})$, $T_2 = CMAC(T_1, ID_A||ID_B||N_M||N_B||SSS, 64)$, and $T_3 = CMAC(T_1, ID_B||ID_A||N_B||N_M||SSS, 64)$. \mathcal{M} sends $\{ID_B||ID_A||SSS||AC||N_M||PK_{MX}||PK_{MY}||T_3\}$ to \mathcal{B} . \mathcal{M} computes $T_4 = LMB_{128}(DH_{Key})$, and generates the master key $MK = CMAC(T_4, N_M||N_B, 128)$.
- \mathcal{B} verifies that $T_3 = T'_3$, computes $T'_4 = LMB_{128}(DH_{Key})$, and generates the master key $MK = CMAC(T'_4, N_M||N_B, 128)$.

\mathcal{M} and \mathcal{B} reach to the same MK at the end. \mathcal{M} could successfully impersonate \mathcal{A} . A similar scenario for an impersonation attack can be written where \mathcal{M} impersonates \mathcal{B} in communication with \mathcal{A} .

Lack of Forward Secrecy: Here we show that Protocol I does not provide the forward secrecy, and then does not provide the weak-PFS or PFS:

- Assume that SK_B has been compromised. \mathcal{E} , that has eavesdropped and saved all the messages exchanged through previous runs of the protocol, knows PK_A , N_A and N_B . \mathcal{E} computes $DH_{Key} = X(SK_B \times PK_A)$, $T'_4 = LMB_{128}(DH_{Key})$, and obtains the established key $MK = CMAC(T'_4, N_A || N_B, 128)$.
- If SK_A has been compromised, \mathcal{E} computes $DH_{Key} = X(SK_A \times PK_B)$, $T_4 = LMB_{128}(DH_{Key})$, and obtains $MK = CMAC(T_4, N_A || N_B, 128)$.

4.2 Protocol II

Protocol II requires out-of-bank transfer of a node's public key to the hub. It is vulnerable to the KCI attack, and lacks the forward secrecy.

Key Compromise Impersonation Attack: Protocol II is vulnerable to the KCI attack. Here is the attack scenario in which \mathcal{M} has SK_A and impersonates \mathcal{B} . As the public key of \mathcal{B} is sent in clear, we can assume that \mathcal{M} has obtained PK_B by eavesdropping a previous protocol run.

- \mathcal{A} selects a 128-bit random number N_A , and sends $\{ID_B || ID_A || SSS || AC || N_A || 0 || 0 || XX\}$ to \mathcal{B} . \mathcal{M} hijacks the session, and tries to impersonate \mathcal{B} .
- \mathcal{M} selects a 128-bit random number N_M , and sends $\{ID_A || ID_B || SSS || AC || N_M || PK_{BX} || PK_{BY} || XX\}$ to \mathcal{A} .
- \mathcal{M} has SK_A . \mathcal{M} computes $DH_{Key} = X(SK_A \times PK_B)$, $T'_1 = RMB_{128}(DH_{Key})$, $T'_2 = CMAC(T'_1, ID_A || ID_B || N_A || N_M || SSS, 64)$, and $T'_3 = CMAC(T'_1, ID_B || ID_A || N_M || N_A || SSS, 64)$. \mathcal{M} sends $\{ID_A || ID_B || SSS || AC || N_M || PK_{BX} || PK_{BY} || T'_2\}$ to \mathcal{A} .
- \mathcal{A} computes $DH_{Key} = X(SK_A \times PK_B)$, $T_1 = RMB_{128}(DH_{Key})$, and $T_2 = CMAC(T_1, ID_A || ID_B || N_A || N_M || SSS, 64)$. \mathcal{A} verifies that $T_2 = T'_2$, and computes $T_3 = CMAC(T_1, ID_B || ID_A || N_M || N_A || SSS, 64)$. \mathcal{A} sends $\{ID_B || ID_A || SSS || AC || N_A || 0 || 0 || T_3\}$ to \mathcal{M} .
- \mathcal{A} computes $T_4 = LMB_{128}(DH_{Key})$, and generates the master key $MK = CMAC(T_4, N_A || N_M, 128)$.
- \mathcal{M} computes $T'_4 = LMB_{128}(DH_{Key})$, and generates the master key $MK = CMAC(T'_4, N_A || N_M, 128)$.

\mathcal{M} and \mathcal{A} compute the same MK . \mathcal{M} could successfully impersonate \mathcal{B} .

Lack of Forward Secrecy: Protocol II does not provide the forward secrecy. As it is assumed that PK_A has been securely shared with \mathcal{B} , we just consider the case that SK_A has been compromised. We show how \mathcal{E} can extract previously established MK from the eavesdropped messages which proves lack of forward secrecy and PFS: As PK_B , N_A and N_B are sent in clear, we can assume that they are eavesdropped and saved by \mathcal{E} . \mathcal{E} computes $DH_{Key} = X(SK_A \times PK_B)$, $T_4 = LMB_{128}(DH_{Key})$, and obtains $MK = CMAC(T_4, N_A || N_B, 128)$.

4.3 Protocol III

Protocol III is a PAKE protocol. It is vulnerable to impersonation and offline dictionary attacks. It does not provide the forward secrecy.

Impersonation Attack: For performing an impersonation attack to Protocol III, \mathcal{M} first eavesdrops messages between \mathcal{A} and \mathcal{B} in a protocol run. \mathcal{M} then obtains PK'_A and PK_A from messages (1) and (4) of the protocol. \mathcal{M} computes $Q' = PK_A - PK'_A$, and uses Q' for an impersonation attack. Note that we have $Q' = Q(PW)$. Here is an impersonation attack on protocol III, in which \mathcal{M} impersonates \mathcal{A} :

- \mathcal{M} selects a private key SK_M , and generates the corresponding public key as $PK_M = (PK_{MX}, PK_{MY}) = SK_M \times G$. \mathcal{M} computes $PK'_M = PK_M - Q'$. If $PK'_M = O$, \mathcal{M} selects a new private and public key and continues the process until $PK'_M \neq O$. \mathcal{M} selects a 128-bit random number N_M , and sends $\{ID_B||ID_A||SSS||AC||N_M||PK'_{MX}||PK'_{MY}||XX\}$ to \mathcal{B} .
- \mathcal{B} selects a 128-bit random number N_B , and sends $\{ID_A||ID_B||SSS||AC||N_B||PK_{BX}||PK_{BY}||XX\}$ to \mathcal{M} .
- \mathcal{B} calculates $PK_M = PK'_M + Q(PW)$, and computes $DH_{Key} = X(SK_B \times PK_M)$, $T'_1 = RMB_{128}(DH_{Key})$, $T'_2 = CMAC(T'_1, ID_A||ID_B||N_M||N_B||SSS, 64)$, and $T'_3 = CMAC(T'_1, ID_B||ID_A||N_B||N_M||SSS, 64)$. \mathcal{B} sends $\{ID_A||ID_B||SSS||AC||N_B||PK_{BX}||PK_{BY}||T'_2\}$ to \mathcal{M} .
- \mathcal{M} computes $DH_{Key} = X(SK_M \times PK_B)$, $T_1 = RMB_{128}(DH_{Key})$, $T_2 = CMAC(T_1, ID_A||ID_B||N_M||N_B||SSS, 64)$, and $T_3 = CMAC(T_1, ID_B||ID_A||N_B||N_M||SSS, 64)$. \mathcal{M} sends $\{ID_B||ID_A||SSS||AC||N_M||PK_{MX}||PK_{MY}||T_3\}$ to \mathcal{B} . \mathcal{M} computes $T_4 = LMB_{128}(DH_{Key})$, and generates the master key $MK = CMAC(T_4, N_M||N_B, 128)$.
- \mathcal{B} verifies that $T_3 = T'_3$, computes $T'_4 = LMB_{128}(DH_{Key})$, and generates the master key $MK = CMAC(T'_4, N_M||N_B, 128)$.

\mathcal{M} and \mathcal{B} reach to the same MK at the end. \mathcal{M} could successfully impersonate \mathcal{A} . A similar scenario for an impersonation attack can be written where \mathcal{M} impersonates \mathcal{B} in communication with \mathcal{A} .

Offline Dictionary Attack: Protocol III is a PAKE protocol with two-factor authentication. It requires both public keys and a shared password. For PAKE protocols, it is crucial to provide resilience to offline dictionary attacks. If an adversary could guess a password, he should not be able to verify his guess offline. For performing a dictionary attack on protocol III, it is sufficient that \mathcal{E} eavesdrops messages between \mathcal{A} and \mathcal{B} in a protocol run. \mathcal{E} then obtains PK'_A and PK_A from messages (1) and (4) of the protocol. \mathcal{E} computes $PK_A - PK'_A = Q(PW) = (Q_X, Q_Y)$. As $Q_X = 2^{32}PW + M_X$ and Q_X is known, this can be used as a verifier. \mathcal{E} can then try probable passwords from a dictionary of most

probable passwords, and check which password PW will map to Q_X . This can be done very fast, and \mathcal{E} can find the password PW that is shared between \mathcal{A} and \mathcal{B} .

Lack of Forward Secrecy: Protocol III does not provide the forward secrecy. As PK_B , N_A and N_B are sent in clear, we can assume that they are eavesdropped and saved by \mathcal{E} . If SK_A is compromised, \mathcal{E} computes $DH_{Key} = X(SK_A \times PK_B)$, $T_4 = LMB_{128}(DH_{Key})$, and obtains the master key $MK = CMAC(T_4, N_A || N_B, 128)$.

4.4 Protocol IV

Protocol IV is vulnerable to an impersonation attack, and lacks the forward secrecy.

Impersonation Attack: Here is an impersonation attack on protocol IV, in which \mathcal{M} impersonates \mathcal{A} :

- \mathcal{M} selects a private key SK_M , and generates the corresponding public key as $PK_M = (PK_{MX}, PK_{MY}) = SK_M \times G$. \mathcal{M} selects a 128-bit random number N_M , and computes $W_M = CMAC(N_M, ID_A || ID_B || PK_{MX} || PK_{MY}, 128)$. \mathcal{M} sends $\{ID_B || ID_A || SSS || AC || W_M || PK_{MX} || PK_{MY} || XX\}$ to \mathcal{B} .
- \mathcal{B} selects a 128-bit random number N_B , and sends $\{ID_A || ID_B || SSS || AC || N_B || PK_{BX} || PK_{BY} || XX\}$ to \mathcal{M} .
- \mathcal{B} computes $DH_{Key} = X(SK_B \times PK_M)$, $T'_1 = RMB_{128}(DH_{Key})$, $T'_2 = CMAC(T'_1, ID_A || ID_B || W_M || N_B || SSS, 64)$, and $T'_3 = CMAC(T'_1, ID_B || ID_A || N_B || W_M || SSS, 64)$. \mathcal{B} sends $\{ID_A || ID_B || SSS || AC || N_B || PK_{BX} || PK_{BY} || T'_2\}$ to \mathcal{M} .
- \mathcal{M} computes $DH_{Key} = X(SK_M \times PK_B)$, $T_1 = RMB_{128}(DH_{Key})$, $T_2 = CMAC(T_1, ID_A || ID_B || W_M || N_B || SSS, 64)$, and $T_3 = CMAC(T_1, ID_B || ID_A || N_B || W_M || SSS, 64)$. \mathcal{M} sends $\{ID_B || ID_A || SSS || AC || N_M || PK_{MX} || PK_{MY} || T_3\}$ to \mathcal{B} .
- $Display_M$ will show $BS2DI(D)$ in which $D = CMAC(N_M || N_B, N_B || N_M || T_1, 16)$.
- \mathcal{B} verifies that $T_3 = T'_3$, computes $W'_M = CMAC(N_M, ID_A || ID_B || PK_{MX} || PK_{MY}, 128)$, and verifies that $W_M = W'_M$. $Display_B$ will show $BS2DI(D')$ where $D' = CMAC(N_M || N_B, N_B || N_M || T_1, 16)$.
- As $Display_M = Display_B$, \mathcal{B} computes $T'_4 = LMB_{128}(DH_{Key})$ and $MK = CMAC(T'_4, N_M || N_B, 128)$. \mathcal{M} computes $T_4 = LMB_{128}(DH_{Key})$ and $MK = CMAC(T_4, N_M || N_B, 128)$.

\mathcal{M} and \mathcal{B} compute the same MK . \mathcal{M} could successfully impersonate \mathcal{A} . A similar scenario can be written for an impersonation attack where \mathcal{M} impersonates \mathcal{B} in communication with \mathcal{A} .

Lack of Forward Secrecy: Protocol IV does not provide the forward secrecy. As PK_A , PK_B , N_A and N_B are sent in clear, we can assume that they are eavesdropped and saved by \mathcal{E} .

- If SK_B has been compromised, \mathcal{E} computes $DH_{Key} = X(SK_B \times PK_A)$, $T'_4 = LMB_{128}(DH_{Key})$, and obtains $MK = CMAC(T'_4, N_A || N_B, 128)$.
- If SK_A has been compromised, \mathcal{E} computes $DH_{Key} = X(SK_A \times PK_B)$, $T_4 = LMB_{128}(DH_{Key})$, and obtains $MK = CMAC(T_4, N_A || N_B, 128)$.

5 Conclusion

The security of the IEEE 802.15.6 standard for WBAN [3] has been challenged in this paper. We analyzed the security of four key agreement protocols that are used for establishing a master key in the security association process of the standard. We showed that all four protocols have security problems. They are vulnerable to the KCI attack, and lack the forward secrecy. Furthermore, the first, third and fourth protocols are vulnerable to the impersonation attack. The third protocol is also vulnerable to the offline dictionary attack. Further attacks will be feasible if public keys are not validated. The standard aims to provide the confidentiality, authentication, integrity, privacy protection and replay defence. However, our attacks show that the confidentiality and authentication are not achieved by the current security mechanisms in the standard.

Acknowledgement. The author would like to thank Øyvind Ytrehus and the anonymous reviewers for their comments.

References

1. Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., Leung, V.C.: Body area networks: a survey. *Mob. Netw. Appl.* **16**(2), 171–193 (2011)
2. Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., Jamalipour, A.: Wireless body area networks: a survey. *Commun. Surv. Tutorials, IEEE* **16**(3), 1658–1686 (2014)
3. Association, T.I.S.: IEEE P802.15.6-2012 Standard for Wireless Body Area Networks (2012). <http://standards.ieee.org/findstds/standard/802.15.6-2012.html>
4. Krawczyk, H.: HMQV: a high-performance secure diffie-hellman protocol. In: Shoup, V. (ed.) *CRYPTO 2005*. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
5. Menezes, A.: Another look at HMQV. *Math. Cryptology JMC* **1**(1), 47–64 (2007)
6. Toorani, M.: On continuous after-the-fact leakage-resilient key exchange. In: Proceedings of the 2nd Workshop on Cryptography and Security in Computing Systems (CS2 2015), ACM (January 2015)
7. Toorani, M.: Cryptanalysis of a new protocol of wide use for email with perfect forward secrecy. *Secur. Commun. Netw.* **8**(4), 694–701 (2015)
8. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)

9. Toorani, M., Beheshti, A.: A directly public verifiable signcryption scheme based on elliptic curves. In: Proceedings of the 14th IEEE Symposium on Computers and Communications (ISCC 2009), pp. 713–716 (2009)
10. Hankerson, D., Vanstone, S., Menezes, A.J.: Guide to Elliptic Curve Cryptography. Springer, Berlin (2004)
11. Misra, S., Goswami, S., Taneja, C., Mukherjee, A.: Design and implementation analysis of a public key infrastructure-enabled security framework for ZigBee sensor networks. International Journal of Communication Systems (2014)
12. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
13. Toorani, M., Beheshti, A.: Cryptanalysis of an elliptic curve-based signcryption scheme. Int. J. Netw. Secur. **10**(1), 51–56 (2010)
14. Toorani, M., Beheshti, A.: LPKI-a lightweight public key Infrastructure for the mobile environments. In: Proceedings of the 11th IEEE International Conference on Communication Systems(ICCS 2008), pp. 162–166, November 2008. doi:[10.1109/ICCS.2008.4737164](https://doi.org/10.1109/ICCS.2008.4737164)

Visual Cryptography and Obfuscation: A Use-Case for Decrypting and Deobfuscating Information Using Augmented Reality

Patrik Lantz^{1,2(✉)}, Bjorn Johansson¹, Martin Hell², and Ben Smeets^{1,2}

¹ Ericsson Research, Lund, Sweden

{patrik.lantz,bjorn.a.johansson,ben.smeets}@ericsson.com

² Department of Electrical and Information Technology,

Lund University, Lund, Sweden

{patrik.lantz,martin.hell,ben.smeets}@eit.lth.se

Abstract. As new technologies emerge such as wearables, it opens up for new challenges, especially related to security and privacy. One such recent technology is smart glasses. The use of glasses introduces security and privacy concerns for the general public but also for the user itself. In this paper we present work which focus on privacy of the user during authentication. We propose and analyze two methods, visual cryptography and obfuscation for protecting the user against HUD and camera logging adversaries as well as shoulder-surfing.

Keywords: Visual cryptography · Visual obfuscation · Augmented reality · Wearables

1 Introduction

Recent research [1,2] in privacy-preserving human-computer interaction allows users to authenticate and decipher data using smart glasses equipped with a camera. Decrypted data or one-time authorization codes (OTAC) are displayed as an image overlay in a heads-up display (HUD). The user can then interact with a terminal screen while preventing shoulder-surfing as an adversary cannot observe the HUD. However, this does not mitigate attacks where the adversary has access to the information presented in the HUD.

In our proposed methods, using visual obfuscation and a modified visual cryptography scheme, we split information shown in the HUD into two or three partitions. These partitions are displayed on a terminal screen and in the HUD. Decrypting and deobfuscating information is then a matter of aligning the image overlay in the HUD with the information displayed on the screen. For the attack model we assume that an adversary has access to (a) one of two or (b) two of three partitions. The adversary could be shoulder-surfing or is capable of observing the HUD. In case (b) we assume that the adversary is capable of combining these partitions easily. In case of an adversary which has control over the camera, then we rely on (b) only. However, case (a) still holds if camera recording can be disabled or prevented to record [3].

2 Related Work

Earlier work has focused on private interactions and authentications between a user with smart glasses and a terminal screen, protecting against a shoulder-surfing.

Forte et al. present EyeDecrypt [1] which allows an authorized user to decipher data shown on a display. The decrypted data is shown on a mobile device or smart glasses as an image overlay. If the overlay displays digits in a PIN pad, then using augmented reality, the user can securely enter input onto a screen if the user interface is randomized.

In Ubic [2], Simkin et al. describe an authentication protocol involving smart glasses. Users approach a terminal screen and decode a signed visual encoding (such as QR codes). Then the user initiates the protocol by sending his identification to the terminal host which checks the public key of the identifier. The terminal host creates a challenge, encrypted with the public key and displays it on the screen. The user decodes yet another visual encoding, decrypts it and the challenge is displayed in form of an OTAC in the HUD of the glasses which the user enters on a keypad. Simkin et al. shows that the protocol protects against shoulder-surfing but also against active attackers, such as a man-in-the-middle attack.

Previous work on visual cryptography which this paper is influenced by is the Naor-Shamir visual cryptography scheme [4]. In this secret sharing scheme 2 or n users can mechanically decrypt a visual image by overlaying the shares of the images, assuming transparency in the shares. A secret image is broken up into n shares so that the original image will only be decrypted by someone with possession of all the shares.

3 Visual Cryptography

3.1 Original Version

The original idea of Naor-Shamir visual cryptography scheme uses two components created as a number of black and white sub-pixels. These two components are superimposed to reveal the original image. Using a one-time-pad (OTP) with the same size as the original image as the first component and creating an encrypted image by taking the XOR of the original image and the OTP is well known. In order to create a XOR visually each pixel in the original image is represented by a pair of, or 4 sub-pixels and the superimpose is performed by pixel-wise addition. This creates an image which has all white sub-pixels where the original image was 1 and half white/half black where the original image was 0.

3.2 Modified Version

Now we consider the application where a number of secret digits (or characters) are to be shown to a user, e.g. an OTAC or a randomized PIN pad. The digit

information is split into two (or more) parts shown on a terminal screen and a HUD. Instead of having two components of the same type consisting of black and white sub-pixels as in [4], we have one component (screen) consisting of black and white sub-pixels, and one component (HUD) consisting of white and transparent sub-pixels. The HUD dominates the screen meaning that a white pixel in the HUD will make the corresponding pixel in the superimposed image white regardless of the value on the screen for this pixel. For a pixel position that is transparent in the HUD the superimposed image will get the value on the screen for this position. This can be formulated as follows; If we represent black/transparent as 0 and white as 1 we superimpose by pixel-wise OR, or the max-operation in the case of grayscale images as in our experiments.

The creation of the encrypted components is performed as follows, also shown in Fig. 1. First we create a temporary OTP with the same size as the original image consisting of ones and zeros. We then represent a one in the temporary OTP by four sub-pixels in a new OTP with white subpixels on the diagonal $A = \begin{pmatrix} W & T \\ T & W \end{pmatrix}$ and a zero by two transparent pixels on the diagonal $B = \begin{pmatrix} T & W \\ W & T \end{pmatrix}$. This larger image is now used as our OTP. We then create an encrypted original image with the following rule, assuming black digits on white background. If original image pixel is white and OTP is A or original image pixel is black and OTP is B then let the encrypted pixel be represented by $C = \begin{pmatrix} B & W \\ W & B \end{pmatrix}$, otherwise represented by $D = \begin{pmatrix} W & B \\ B & W \end{pmatrix}$. This way the black pixels on the screen are placed so that they are covered by white pixels in the glasses when we want to create a white pixel in the superimposed image, but placed to be seen through the transparent pixels in the glasses when we want to create a black pixel in the superimposed image. This corresponds to creating the encrypted image by taking the exclusive or (XOR) of the original pixel value and OTP.

We use the described technique to encrypt digits which are visually revealed when the two components are superimposed. Figure 2(a) shows a randomized OTP where to each pixel in the original image we have created a 2×2 matrix of sub-pixels. Figure 2(b) shows a picture of the encrypted data visualized on a computer screen captured by a camera. For each pixel in the original image a 2×2 pattern for the encrypted image was created according to the above. Estimated camera parameters were used to compensate for radial and tangential distortions in the picture. The transformation used for warping the OTP to match the picture was estimated manually but could be estimated automatically

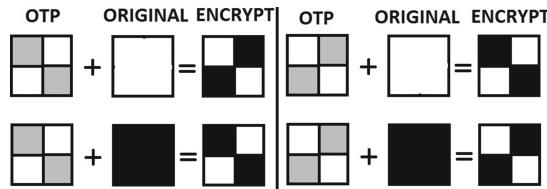


Fig. 1. Encryption by superimposing OTP

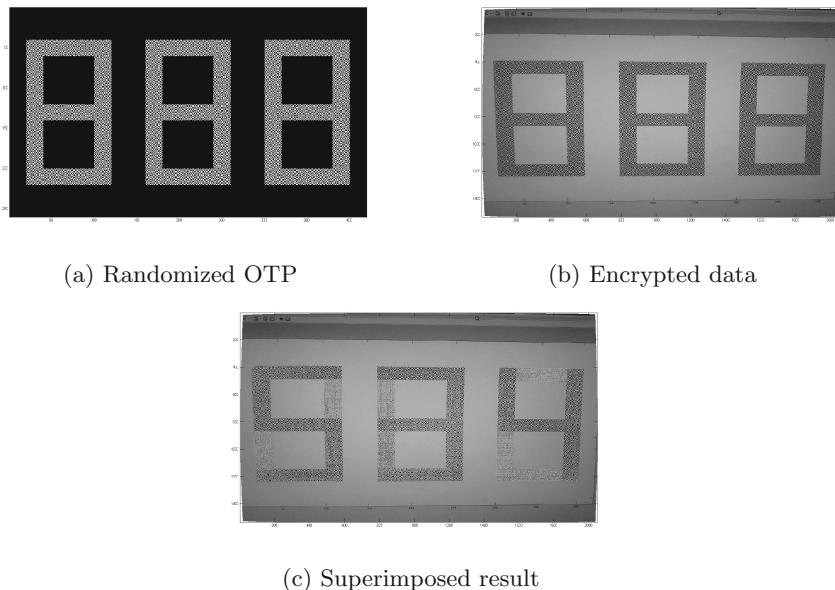


Fig. 2. Encrypting digits

using standard techniques in computer vision, alternatively the screen and head could be rotated and tilted so that the components match. The superimposed result is shown in Fig. 2(c) where the original text can be seen. Due to the following sources for errors the visual decryption is not perfect

- Image distortions due to imperfect camera (e.g. nonlinearities)
 - ‘Bleeding’ of white areas into black in picture smoothing OTP
 - Estimated transformation does not perfectly warp OTP to picture

3.3 Using a Seven-Segment Display

Next we restrict the visual representation of the digits to the well known digital font consisting of seven bars. In this case we can use two sub-bars for each line-segment in the font, similar to what is described by Bochert [5]. In the OTP and in the encrypted image one of the two bars for each line-segment is set, white sets in Fig. 3(a) (black background is transparency) and black sets in Fig. 3(b) respectively. For the line-segments that are set in the original image to create the digit, the OTP and encrypted image will have different sub-bars set. For the line-segments that are not set in the original image, the line-segments in the OTP and the encrypted image will have the same sub-bars set and will cancel out each other. In the superimposed image the digit will appear in the clear, see Fig. 3(c). Note that in the example below the OTP sub-bars are created larger than in the encrypted image, in order to make the system less sensitive to the

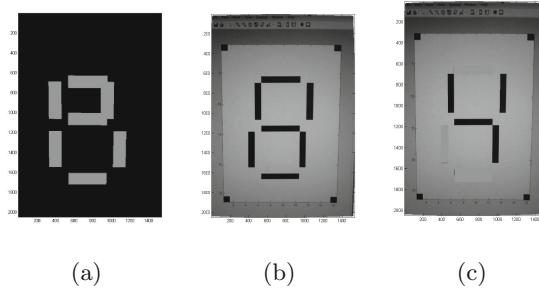


Fig. 3. Using two sub-bars for each line-segment

errors mentioned before. Compared to the approach above, this approach is less sensitive to the alignment between the OTP and the encrypted image but may be visually less attractive.

4 Visual Obfuscation

In this next section we describe a method for obfuscating digits. As opposed to the visual cryptography scheme that might suffer from difficulties of alignment the proposed method is not as sensitive. However, in this scenario the attacker has a higher probability to guess correct digit compared to the visual encryption where no information about the digit is learned from by observing one of the partitions.

4.1 Digit Representation

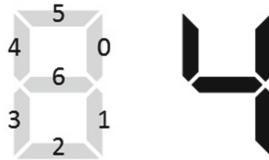
For the obfuscation case we represent the digits using seven-segment fonts. This font can easily be divided into several partitions that are shared among a number of digits. The more digits that share the same partitions, the harder it is to guess the correct digit. Figure 4 shows the bars numbered 0 to 6 that can form all possible digits zero to nine. The digits can be encoded using the binary sequence $x_0x_1\dots x_6$ where

$$x_i = \begin{cases} 1, & \text{if bar } i \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

As an example, digit four in Fig. 4 is encoded as the sequence 1100101. The full table of encodings is shown in Table 1.

4.2 Analysis of 2-Way Partitioning

Each of the digits shown to the user are partitioned into two parts, one of the partitions are shown in the HUD of the glasses and the other partitions are located on the terminal screen. The user aligns the image overlay in the HUD

**Fig. 4.** Bar numberings for binary encoding**Table 1.** List of binary encodings and number of partitions for $n \geq 2$

Digit	Binary encoding	# of partitions
1	1100000	2
2	1011011	16
3	1110011	16
4	1100101	8
5	0110111	16
6	0111111	32
7	1100010	4
8	1111111	48
9	1100111	16
0	1111110	32

with the partitions shown on the screen in order to deobfuscate. If an adversary gets access to one of the two partitions, the probability to guess correct digit should be low.

The number of combinations a digit can be partitioned into two parts in is shown in Table 1. Note that one partition can be displayed in the HUD and the other partition on the screen or vice versa which make the actual number of possible partitions twice the number shown in Table 1. In general, the number of partition combinations should be m^i where m is number of partitions and i is number of ones in the binary encoding needed to represent a digit. We only consider partitions which could potentially be used by more than one digit. Therefore, for the digit eight the value should be $2^7/2$ partitions but instead it is 48. This is due to the fact that in some partitions, one of the parts reveals the whole digit, leaving no other candidates to choose from except the digit eight.

Each partition can be part of n number of candidate digits giving a naïve attacker a chance of $1/n$ to guess the correct digit by observing only one partition.

However, the probability that a particular part is derived from a particular digit is not the same for all digits so an analysing attacker can do much better. Assume all digits have the same probability and we have an equal probability distribution for the partitions of each digit. The best strategy for an attacker in this case is to guess on the digit having the largest probability for the observed

partition. As an illustrative example we choose the digit 4 and partition 5 from Table 2. In this case the adversary would guess number 4 with the probability to guess correct equal to $\Pr(4 \mid 0000100) = 0.3808$ using the values in Table 3. On the other hand, if the adversary has access to the second partition the probability becomes $\Pr(4 \mid 1100001) = 0.470$. We calculated the mean probability to guess one digit correct observing one partition to be 0.45 when observing only one part and assuming equally probable digits and equal distribution among the partitions for each digit. In Sect. 4.3 we elaborate on the details for how this is computed.

From simulations we also conclude that the mean number of possible PIN pad solutions that match an observed partition of a 10-digit PIN pad is approximately 1300. The number of solutions s for *one* PIN pad with one partition for each digit drawn randomly with equal probability is computed using $s = |S|$. S is a set of 10-tuples with distinct elements and is given by

$$S = \{t_j \mid a_k \neq a_l, \forall a_k, a_l \in t_j \wedge k \neq l\}$$

where $t_j \in T$, for $j = 1, 2, \dots, |T|$ and

$$T = D_{p_0} \times D_{p_1} \times \dots \times D_{p_9}$$

p_i denotes the partition at index i in the PIN pad and D is a set of possible digits that a partition p_i can form. In the simulations we compute the average number of solutions for a large number of random PIN pads using s . This gives a naïve attacker about 1 in 1300 to guess the PIN pad correct. Note that the attacker only has to guess the pressed buttons correct in order to have the PIN though.

Table 2. All possible partitions of digit 4 as described in Table 1 and how many possible digits each part could originate from

Partition	Part 1	Part 2	Possible digits from part 1	Possible digits from part 2
1	0000000	1100101	10	3
2	1000000	0100101	8	5
3	0100000	1000101	9	3
4	1100000	0000101	7	5
5	0000100	1100001	6	4
6	1000100	0100001	4	6
7	0100100	1000001	6	5
8	1100100	0000001	4	7

Table 3. Probabilities for the two parts of partition number 5 from Table 2 being part of a particular digit

Partition	1	2	3	4	5	6	7	8	9	0
0000100	0	0	0	0.0625	0.0313	0.0156	0	0.0078	0.0313	0.0156
1100001	0	0	0.0313	0.0625	0	0	0	0.0078	0.0313	0

4.3 Optimizing the Partitioning

The equal probability distribution for the different parts a digit can be partitioned into as described above may not be the optimal choice if we want to minimize the probability for an adversary to guess correct digit when observing only one part of the partitioning. Let a_{ij} be the probability that digit D_j is chosen to be partitioned and that part P_i is used as one partition. We represent the distribution in a 128×10 matrix A where entry (i, j) is denoted a_{ij} . Only 380 out of 1280 entries in A have non-zero entries since most partitions cannot be used to create a particular digit. Note that Table 3 consist of two rows from A .

$$A = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{09} \\ a_{10} & a_{11} & \dots & a_{19} \\ \vdots & \vdots & \ddots & \vdots \\ a_{127,0} & & \dots & a_{127,9} \end{bmatrix}$$

First we assume that an attacker knows the distributions of the digits and partitions, that is the matrix A . The optimal strategy for the attacker is to guess the digit having the highest value among the observed partition. The probability of a correct guess when observing partition i is equal to

$$\frac{\max_j a_{ij}}{\sum_j a_{ij}}$$

and the mean probability to guess correct on any partition is equal to $\sum_i \max_j a_{ij}$. In order to minimize the probability for a successful attack we want to minimize this expression over all a_{ij} . There are constraints on A to be valid though. In some applications the probability for each digit should be equal and the probability for all digits should sum up to 1 which gives the constraint $\sum_i a_{ij} = 0.1$, for $j = 1, \dots, 10$. Secondly, the two partitions that form the digit must have the same probability, $a_{kj} = a_{lj}$ if partition P_k and partition P_l form digit D_j . Our optimization problem can then be expressed as follows:

$$\begin{aligned} & \text{minimize} \quad \sum_i \max_j a_{ij} \\ & \text{subject to:} \quad \sum_i a_{ij} = 0.1, \quad j = 1, \dots, 10 \end{aligned}$$

$a_{kj} = a_{lj}$ if P_k and P_l form D_j , $a_{ij} = 0$ if P_i is not part of D_j

This optimization problem is not linear, but can be made linear by the following trick. By introducing helper variables x_0, \dots, x_{127} one for each maximum, adding constraints $x_0 \geq a_0, \dots, x_0 \geq a_9, x_1 \geq a_{10}, \dots, x_{127} \geq a_{127,9}$ and change the expression to minimize to $\sum_i x_i$ we get a linear optimization problem that can be solved e.g. using linear programming (LP). We have solved the optimization problem above and the optimal solution gives the attacker a chance of 0.3743 to guess the correct digit observing only one part and knowing which distribution is used when splitting the digits.

A variant of the conditions above is that we do not require the digits to have equal probability. A possible scenario could be when we want to show an OTAC consisting of a number of digits to a user by partitioning it in two parts. If you want to minimize the chance for an attacker to guess the digits in the OTAC you might want to use the digits that are easier to guess (e.g. 1) less frequently than digits that are harder to guess. The first constraint from above is then substituted for

$$\sum_{ij} a_{ij} = 1$$

A simple distribution that fulfills the constraints is to let all a_{ij} be equal, in this case $a_{ij} = 1/380 \forall i, j$. This gives a probability of guessing correct 0.2947. If we use LP to determine the global optimum given this new constraint we get a probability of 0.2867 for an attacker to guess correct if he knows the distribution.

In the analysis above we assume that the attacker knows the distribution matrix and uses an optimal strategy for guessing. In reality we may use different distributions each time. If it is known that the attacker uses the equally distributed matrix B in the guessing we could create other distributions that make the attacker less successful. Also the problem to create such a distribution that minimizes the probability for a successful attack can be expressed as a linear optimization problem and solved e.g. using LP. Let b_{ij} be entries in the known distribution matrix B the attacker uses. The optimization problem can then be expressed as

$$\min \sum_{ij} c_{ij} \cdot a_{ij} \text{ over } a_{ij}$$

$$a_{kj} = a_{lj}, \text{ if } P_k \text{ and } P_l \text{ form } D_j$$

$$a_{ij} = 0 \text{ if } P_i \text{ is not part of } D_j$$

where c_{ij} is constructed according to

$$c_{ij} = 0 \text{ if } b_{ij} = 0$$

$$c_{ij} = 1/d \text{ if } b_{ij} = \max_j b_{ij} \text{ and } d = \text{number of max on row } j$$

Using a distribution created this way an attacker guessing according to the equal probability distribution will only have a probability of 0.2833 to guess correct.

4.4 Analysis of 4-Bar Shape

The main reason the attacker has a rather high probability to guess a digit correct is that the font we use for the digits has 7 bars to represent only 10 digits. If we instead should use the minimal number of bars (four) it would be harder for an attacker to guess correct. Then we would not recognize the representation as our traditional digits but it is interesting to compare the probabilities for this case with our earlier results.

We use 4-bar shapes using the binary encodings $x_0x_1\dots x_3$ and choose 10 shapes among the 15 possible shapes. Similar calculations as for the 2-way partition analysis were performed for the 4-bar shapes in which the probability becomes 0.2667 instead to guess the correct shape. This result shows that using fewer bars improves this obfuscation method. If we compare this to a two digit OTAC where we have one digit in clear and require to guess the other one, the probability is 0.10 compared with using four 4-bar shapes in which the probability is 0.071.

An alternative way to minimize the number of bars is to let the 7 bars represent more figures. Besides the traditional digits we include a number of characters that can be created from the used digital font. These can be chosen from the set A,C,E,F,G,H,J,L,P,U. We found that the optimal distributions for the scenario with 10 digits and 8 characters, an attacker who knows the distribution has probability 0.2479 to guess correct for equal probable digits/characters and 0.1753 to guess correct for non-equal probable digits/characters. We did the same analysis for the case where we use all 127 figures that can be made up from the seven bars for comparison. In Sect. 5 we compare the results for the different scenarios from above and discuss the conclusions from them.

4.5 Analysis of 3-Way Partitioning

In our worst-case scenario, the attacker has access to the HUD display and has hijacked the camera in order to combine the HUD- and terminal-view and is able to record the user input. In this case, the adversary will always be able to guess the digits by either manual analysis or computational image analysis.

Now we assume that there is an autostereoscopic [6] display available for which the left eye will see different information on the screen than the right eye (camera-view). The obfuscation can be split into three partitions designated for the left and right eye and the HUD.

In our simulations we assume that we do not know what partitions the adversary can observe, it could theoretically be any two. In a simulation similar to the 2-way partitioning we investigate the mean probability for an adversary to guess a digit while having two out of three partitions and combining these for calculating the best candidate digit. The result becomes now 0.60, assuming equal probability of the partitions. If observing only one of three partitions, the probability is 0.28 to guess correct.

5 Results

In this section we describe comparative results of the analysis from Sect. 3.

The probability 0.3743 for using all digits makes the chance to guess a two digit OTAC equal to 0.1401 which is larger than the probability to guess one digit out of 10 (0.10), which would be the case if we instead would have obfuscated using our benchmark case, that is showing half of the digits in cleartext on the screen and remaining digits in cleartext on the HUD. If we use certain digits that are easy to guess less frequently (non-equally probable digits) as we can in the OTAC case, the probability of guessing one digit correct becomes 0.2867, the probability to guess the two digit OTAC is 0.0822. Hence, the attacker is less successful if we use this approach for showing an OTAC compared to the benchmark case.

In the comparison we want to show how long OTACs are needed if we compare against an OTAC of length 100 as the benchmark case. The cases we compare with and the results are shown in Table 4.

Table 4. Comparison of OTAC lengths for different constructions against the benchmark case consisting of 100 digits.

Type	Digits	Description
Benchmark	100	Half of the digits in cleartext on the screen and the remaining digits in the HUD
10Dig-I	118	10 digits, all with equal probability
10Dig-II	93	10 digits with different probability
10Dig10Let-I	106	10 digits and 10 letters, all with equal probability
10Dig10Let-II	85	10 digits and 10 letters with different probability
10Dig8Let-I	102	10 digits and 8 letters, all with equal probability
10Dig8Let-II	83	10 digits and 8 letters with different probability
127Char-I	93	127 characters, all with equal probability
127Char-II	79	127 characters with different distributions
10Dig4Bar-I	83	10 digits represented as 4-bars, all with equal probability
10Dig4Bar-II	79	10 digits represented as 4-bar with different probability
Uniform	91	It is known that the attacker guess according to equal distribution

More specifically, these values show how many digits we can use for the different cases before the probability to guess the digits becomes equal to guessing our 100 digit benchmark. According to our analysis our approach is much better than the benchmark if we use some digits more frequently. Even better results are achieved if we also use characters along with the digits. Note that these results assume that the attacker knows the distribution used for splitting the

digits. If he does not know the distribution, or if it is known what distribution the attacker is using in the guessing, the probability to guess an OTAC correct is even lower.

6 Discussion

In the PIN pad case it is crucial that the same digits are placed in each partition. If we would place digits randomly between the partitions, an attacker that observes multiple sessions will be able to combine the sessions and learn the full PIN pad. The PIN pad can be randomized as the authors also suggest in [1], additionally, similar to their advantages, a keylogger running on the host will not learn anything about the user input if there is a touchscreen present. The main differences are that we prevent the information to be leaked when an adversary can observe the HUD or even perform camera logging while at the same time preventing shoulder-surfing. Instead of splitting information in three partitions we can split it into two partitions and disable the camera as our method does not rely on a camera. The user can align the glasses himself to the screen by moving and tilting the head. However, the camera might need to be activated during initial interaction with the terminal needed for any necessary setup between the glasses and the terminal, but after this it could be deactivated.

It is also worth mentioning that the visual cryptography scheme can also be split into three partitions. In this case, the encrypted data can be shown on an autostereoscopic screen while the OTP is displayed in the HUD.

Future work involves mainly to achieve better alignment for the visual cryptography scheme and investigate the usability of both methods by implementing them in a real pair of smart glasses and conducting user-studies. Applying these methods on letters for sensitive text materials seems to be an interesting application as well.

7 Conclusion

In the first method we have adapted the traditional visual cryptography scheme for use with smart glasses. Using this instead of splitting the digits in two parts as described in Sect. 3 has the advantage that the number of bars in the model of the digit has no effect on the probability to guess correct digit if you view only one component. However, it requires a more precise aligning between the two components and is perhaps visually not as attractive.

Our second suggestion is to visually obfuscate digits by partitioning them into two or three distinct parts so that it is hard for the attacker to guess correct when observing only one part. The probability for an attacker to guess correct is dependent on how we chose to split up the digits and what strategy and information the attacker has. We have formalized this and formulated optimization problems to find the best distributions used to split up the digits into parts minimizing the probability for a successful attack. There are different scenarios depending on whether all digits must be equally probable or some digits can be

more frequent, if the attacker knows the distribution used to split up the digits, if digits and characters are used, etc. By converting the nonlinear optimization problem to a linear optimization problem we can be sure that the found optimum is global. The conclusion from the results is that if the digital font with 7 bars is used to represent only the digits with the traditional appearance, all digits are equally probable and the attacker knows the distribution used to split up the digits, then the benchmarking approach to display half of the digits on the screen and the other half in the HUD makes it harder for an attacker to guess correct digit. However, if we are allowed to use some digits more frequently our approach is much better. Even better results can be achieved by also using characters along with the digits and if the attacker does not know the distribution he will be even less successful in guessing correct. These results makes the OTAC application more suitable than the PIN pad for our obfuscation as in the PIN pad all digits must occur once and only once.

References

1. Forte, A.G., Garay, J.A., Jim, T., Vahlis, Y.: EyeDecrypt - private interactions in plain sight. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 255–276. Springer, Heidelberg (2014)
2. Simkin, M., Schröder, D., Bulling, A., Fritz, M.: Ubic: bridging the gap between digital cryptography and the physical world. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part I. LNCS, vol. 8712, pp. 56–75. Springer, Heidelberg (2014)
3. Truong, K.N., Patel, S.N., Summet, J.W., Abowd, G.D.: Preventing camera recording by designing a capture-resistant environment. In: Beigl, M., Intille, S.S., Rekimoto, J., Tokuda, H. (eds.) UbiComp 2005. LNCS, vol. 3660, pp. 73–86. Springer, Heidelberg (2005)
4. Naor, M., Shamir, A.: Visual cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)
5. Borchert, B.: Segment-based visual cryptography, Technical report WSI-2007-04. Wilhelm-Schickard-Institut für Informatik, Tübingen (2007)
6. Dodgson, N.A.: Autostereoscopic 3D displays. IEEE Comput. **38**(8), 31–36 (2005)

Ok Glass, Leave Me Alone: Towards a Systematization of Privacy Enhancing Technologies for Wearable Computing

Katharina Krombholz^{1(✉)}, Adrian Dabrowski¹,
Matthew Smith², and Edgar Weippl¹

¹ SBA Research, Vienna, Austria

{kkatharinakrombholz,aadriandabrowski,eedgarweippl}@sba-research.org

² Usable Security and Privacy Group, University of Bonn, Bonn, Germany

smith@cs.uni-bonn.de

Abstract. In the coming age of wearable computing, devices such as Google Glass will become as ubiquitous as smartphones. Their foreseeable deployment in public spaces will cause distinct implications on the privacy of people recorded by these devices. Particularly the discreet recording capabilities of such devices pose new challenges to consensual image disclosure. Therefore, new *Privacy Enhancing Technologies (PETs)* will be needed to help preserve our digital privacy. At the time of writing, no such PETs are available on the market to communicate privacy preferences towards Glass. In the scientific literature, a handful of approaches has been presented. However, none of them has been evaluated regarding their affordances and overall usefulness. In this paper, we provide the first systematization and qualitative evaluation of state of the art PETs that were designed to communicate privacy preferences towards (wearable) cameras, such as Google Glass. The purpose of this paper is to foster a broader discourse on how such technology should be designed in order to be fully privacy preserving and usable.

1 Introduction

Wearable computers with integrated cameras such as Google Glass might soon become as ubiquitous as smartphones. Due to their hands-free user interface and the discreet recording capabilities, collecting and sharing images and videos becomes easier than ever. In contrary to smartphones and other mobile devices, Google Glass literally remains in the wearer's face all the time. Consequentially, many bystanders view such wearables as invasive and fear substantial implications on their digital privacy. Since the paradigm shift to user-generated content on the Internet, the awareness for picture privacy has risen. The foreseeable deployment of wearable technology in public spaces is about to multiply the set of challenges related to non-consensual disclosure of graphical material on the Internet. In such situations, getting informed consent of all people recorded by such a device is unfeasible. Recently, attacks against Google Glass wearers in public have been reported in the media [1]. These scenarios highlight the high

societal demand for *Privacy Enhancing Technologies (PETs)*. At the time of writing, no PETs are available on the market to communicate privacy preferences towards wearable cameras. In scientific literature, a handful of approaches has been published, however none of them placed an emphasis on whether they are actually useful if deployed in particular situations where users are constrained in what artifacts they can carry or wear. The goal of this paper is to provide a first systematization of PETs that have been published in scholarly articles. To do so, we propose a collection of properties and criteria for categorization. The purpose of this paper is to start a discussion on both design and research directions in the fields of security, privacy and HCI in order to ensure that future PETs successfully counter privacy threats in the upcoming era of wearable computing. Additionally, our systematization provides a first suggestion how a standardized evaluation framework for (wearable) PETs could look like. In the course of our extensive literature review, we found approaches of how future PETs might look like. Most of them however, have substantial limitations such as that they address a narrow scenario, exclude particular user groups or cause further privacy challenges due to their privacy-violating functionality. These limitations may reduce the user's subjective satisfaction and introduce errors. Hence, they potentially have an impact on user experience.

2 Properties

The properties presented in this section haven been selected as a set we believe highlights important evaluation dimensions with respect to usability, in particular subjective user satisfaction, learnability, memorability, errors and efficiency.

User-Initiated: In order to mediate privacy-preferences, some PETs require a user to perform an action. Due to the unobtrusive recording capabilities of wearable cameras, users may be recorded by such a device without actually being aware of it. Therefore, user-triggered mediation hinders the consequent communication of a user's privacy preference. This is very likely to cause errors and misunderstandings which are very likely to have a negative impact on the overall user experience.

Location-Based: As determined by Denning et al. [4], privacy preferences can be determined by certain situations or locations. The definition of a privacy-sensitive space mostly varies from user to user, highly depending on their socio-cultural background. However, most users may require a device that works regardless of a certain location. Therefore, location-dependency may be a limiting factor. As a location-based approach usually requires the transmission of location information to other entities (e.g. a trusted server via a secure channel or another device in the surrounding), new privacy challenges are implied.

Face-Recognition-Based: In order to correlate the user of a PET with a user in an image or video taken by a wearable camera, facial recognition could be an efficient method of choice as state of the art algorithms provide sufficient accuracy to correctly identify individuals. Certainly, facial recognition requires

the transmission of privacy-sensitive data to an associated service and therefore poses significant challenges to preserve the user's privacy.

Visual-Marker-Based: As widely used in augmented reality applications, person-tracking can efficiently be performed using visual markers. For PETs, this means that a user has to carry or wear one or multiple visual markers in order to communicate privacy policies towards wearable cameras. Markers can be designed in an obtrusive or unobtrusive way. In certain situations or cultural groups, subtle markers could be preferred over invasive ones and vice versa.

Gesture-Based: In augmented reality applications, gestures are another feasible approach to track individuals in videos. This approach is mostly limited to long or full-shot videos as it is obviously difficult to perform gesture-recognition on single images or smaller image selections. Furthermore, gestures have to be actively performed and therefore require a user who is aware of being filmed. It furthermore poses distinct accessibility challenges for people with physical disabilities and elderly users.

Signal-Emission-Based: As cameras are sensitive to a certain light spectrum, signal-emitting jamming could be used in PETs. Signal-emission based approaches are mostly expensive as they require a dedicated technical artifact which makes them significantly obtrusive.

Physical-Artifact-Required and/or Dedicated-Device-Required: This is an umbrella property for all approaches that require a dedicated physical artifact in order for the PET to function (e.g. an electronic device or visual markers). From the user's perspective, tangible interfaces as provided by such artifacts offer advantages and disadvantages. The main disadvantage is that the user has to carry or wear the artifact at any time. In some situations, this is unfeasible (e.g. in spaces where digital artifacts cannot be operated due to environmental constraints). In contrary, physical artifacts are often easier to understand for the user and give them a sense of control.

Requires-Trusted-Third-Party-Service: If communication with an external service is required, an Internet connection is indispensable. This might be unfeasible in some scenarios, where users are limited in what devices they can carry (e.g. at a beach)

Smartphone-Based: Smartphone-based approaches are easy to deploy since most individuals in today's society carry one with them all the time. However, situations where smartphones are not applicable but preserving an individual's privacy is required. An example of such a situation would be sunbathing and wearing only a bikini.

Visibility: Some PETs require physical artifacts in order to function. When deployed, some of them are highly visible to bystanders and therefore instantly disclose a certain privacy preference to nearby individuals. In some cultures or even particular situations, a more subtle and unobtrusive technology could be preferred by the user. However, others may want to use an obtrusive PET in

order to disclose their privacy preference or policy openly when recorded by a wearer of Google Glass or a similar device. Low visibility however may constrain the communication of a cognitive model towards the user. It also implies a lack of feedback options.

Accessibility: As digital privacy affects all user groups likewise, a PET should work regardless of disabilities or other physical or cognitive conditions, such as low motor control or visual impairments.

Anonymity: PETs should not imply further privacy violations due to their functionality. Approaches that use facial recognition or location-tracking potentially violate the privacy of their users. Presumably, users of PETs are highly concerned about their privacy and potentially perceive privacy-violating PETs as paradoxical.

Impacts-User-Behavior: Some PETs heavily impact the user's behavior, as they either require a high effort in preparation or require the user to perform an action. PETs that require a user to be aware of being filmed also potentially influence the user's behavior.

Requires-Devices-To-Comply: This property indicates whether a PET can be deployed only if (wearable) cameras are updated accordingly (software and/or hardware).

3 Systematization of Privacy Enhancing Technologies

Table 1 presents our systematization in which we indicate if a certain PET has a certain attribute or not. If a PET could be configured in a way to evoke a certain property, we assume a best-case working scenario supposing that a poor implementation would make any concept potentially unusable. Obviously, some properties are disadvantageous for the overall user experience. For this systematization, we refrain from introducing a rating scheme and prefer a non-judgmental presentation. The reason for this is that neither we nor the authors of the respective PETs conducted user studies that would confirm these assumptions. While some of the PETs presented in this section have been particularly designed for the mobile/wearable computing domain, others were designed to preserve picture privacy in general. For the purpose of fostering a fruitful discourse however, we discuss some potentially impacting factors in a qualitative way.

The **Privacy Makeup** and hair-style approach as presented by Harvey et al. [5] exploits the weaknesses of commonly used face detection systems. To inhibit the feature response of face detection algorithms, significantly invasive distortions are created with camouflage makeup. This approach is time consuming in preparation and visually dominant. It therefore hinders everyday social interaction and can provoke unwanted reactions. It is only feasible when facial recognition algorithms are used or the makeup is applied in a way that its wearer is unrecognizable. For unexperienced users, it is hard to apply the makeup correctly. The **Respectful Cameras** approach as presented in [9] uses colored hats

Table 1. Systematization of PETs

	User-Initiated	Location-Based	Face-Recognition-Based	Visual-Marker-Based	Gesture-Based	Signal-Emission-Based	Dedicated	Physical-Device-Required	Communicates-With-Server	Internet-Connection-Required	Smartphone-Based	Visibility	Accessibility	Anonymity	Impacts-User-Behavior	Requires-Devices-To-Comply
Privacy Makeup [5]			•								•	•	•	•		
Respectful Cameras [9]			•			•				•	•	•	•	•		•
P3F [3]			•			•										•
OfflineTags [7]	•		•			•				•	•	•	•	•		
Privacy Visor [11, 12]				•	•					•	•	•				
SnapMe [6]	•	•					•	•	•							•
FaceBlock [13]	•	•					•	•	•							•
BlindSpot [8]	•				•	•					•	•	•	•		
Place Avoider [10]		•									•	•	•	•		
Privacy Gestures [2]	•			•					•		•	•	•	•		

and scarfs as visual markers. Depending on whether an individual prefers to be made unrecognizable or not, the corresponding artifact is chosen and worn in front of a camera. The Picture-Privacy-Policy framework (**P3F**) as presented in [3] uses a similar approach, however the privacy policies used in this scheme are more complex and fine-grained. The visual markers of the respectful cameras approach [9] is based on a binary privacy policy and obtrusive markers. The P3F use not only dedicated accessories but aims at providing a clothing pattern database with fashionable clothing patterns that are then used as visual markers. A large-scale deployment as presented in the paper, however, would require all cameras or picture publishing platforms to use the P3F software to detect the visual markers and to deduct the privacy policies from them. Another visual-marker based approach is **Offlinetags** [7]. Offlinetags uses four different symbols readable by the open-source Offlinetags software. These symbols can simply be printed on a piece of paper and then presented to a camera. In contrary to the other visual-marker-based approaches presented in this section, the obtrusive markers must be presented actively towards a camera. Yamada et al. [11, 12] presented the **Privacy Visor**, i.e. glasses with infrared light sources that are visible to most camera sensors but invisible to the human eye. The goggles approach requires a constant power supply and infrared LEDs that can keep up with the ambient light. As most portable devices come with GPS sensors, location-based technologies such as the **SnapMe** privacy watchdog [6] or **Blind Spot** [8] are feasible to mediate privacy preferences. These approaches are based on correlated location information of a camera and its bystanders. Additionally to the location-reference, SnapMe proposes the use of facial recognition to

identify individuals in pictures. In comparison to SnapMe, the Blind Spot approach is based on fixed cameras and intended for CCTV-like surveillance systems and thus limited to a specific location. **FaceBlock** [13] is based on biometric features on images taken by a (wearable) camera. Similar to the other facial-recognition-based approaches in this section, the FaceBlock system implies further privacy challenges, as privacy-sensitive biometric information is processed and transferred to a (trusted) server. Both FaceBlock and SnapMe provide a smartphone app where users can configure their privacy-settings. The **PlaceAvioder** [10] approach is not only intended to protect the privacy of bystander but also of the wearer of a wearable camera. Similar to BlindSpot, it provides black-listing of privacy-sensitive spaces like bathrooms and bedrooms. Similar to other location-based approaches, it requires a predefined location and might therefore not be applicable in all desired situations.

Barhm et al. [2] presented a gesture-based method (**Privacy Gestures**) to communicate privacy preferences. Individuals perform defined gestures when recorded by a camera. Even though no additional artifact is required, its feasibility is limited to situations where an individual is aware of being recorded.

4 Conclusion and Work in Progress

As wearables such as Google Glass are very likely to capture private information of individuals recorded by these devices, PETs have become necessary to preserve our digital privacy. In this work, we provide an evaluation of PETs to communicate privacy preferences towards wearable cameras. At the time of writing, no such technology is available on the market. In this work, we assembled and systematized PETs that were mostly published at distinguished scientific conferences. We found that most of them are limited to certain pre-defined scenarios or exclude specific user groups: Smartphone-based approaches exclude smartphone abstainers who might refrain from using smartphones for privacy reasons. Some PETs require the collection and transmission of private information such as the location or biometric features and therefore imply further privacy challenges. The purpose of this work is to initiate a discourse between designers as well as security and usability experts and researchers and to provide the fundamentals for establishing a standard benchmark to evaluate conceptual PETs. Based on the results presented in this paper, we are currently conducting a comprehensive user study with qualitative interviews in the field. Our preliminary results suggest that privacy-aware potential users highly desire fully privacy-preserving tools. Furthermore, we found that some particularly unobtrusive PETs are hard for the user to understand and to use as PETs with low visibility do not sufficiently communicate cognitive models to the user.

Acknowledgements. We would like to thank Johanna Ullrich and the reviewers for their insightful comments. The research was funded by COMET K1, FFG - Austrian Research Promotion Agency.

References

1. Google Glass targeted as symbol by anti-tech crowd. <http://edition.cnn.com/2014/04/14/tech/mobile/google-glass-attack/>. Accessed 10 July 2014
2. Barhm, M.S., Qwasmi, N., Qureshi, F.Z., el-Khatib, K.: Negotiating privacy preferences in video surveillance systems. In: Mehrotra, K.G., Mohan, C.K., Oh, J.C., Varshney, P.K., Ali, M. (eds.) IEA/AIE 2011, Part II. LNCS, vol. 6704, pp. 511–521. Springer, Heidelberg (2011)
3. Dabrowski, A., Weippl, E.R., Echizen, I.: Framework based on privacy policy hiding for preventing unauthorized face image processing. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 455–461. IEEE (2013)
4. Denning, T., Dehlawi, Z., Kohno, T.: In situ with bystanders of augmented reality glasses: perspectives on recording and privacy-mediating technologies. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, pp. 2377–2386. ACM (2014)
5. Harvey, A.: CV Dazzle, 2010–2012. <http://cvdazzle.com/>. Accessed 10 April 2014
6. Henne, B., Szongott, C., Smith, M.: Snapme if you can: privacy threats of other peoples' geo-tagged media and what we can do about it. In: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 95–106. ACM (2013)
7. Pallas, F., Ulbricht, M.-R., Jaume-Palasí, L., Höppner, U.: Offlinetags: a novel privacy approach to online photo sharing. In: CHI 2014 Extended Abstracts on Human Factors in Computing Systems, CHI EA 2014, pp. 2179–2184. ACM, New York (2014)
8. Patel, S.N., Summet, J.W., Truong, K.N.: Blindspot: creating capture-resistant spaces. In: Senior, A. (ed.) Protecting Privacy in Video Surveillance, pp. 185–201. Springer, Heidelberg (2009)
9. Schiff, J., Meingast, M., Mulligan, D.K., Sastry, S., Goldberg, K.: Respectful cameras: detecting visual markers in real-time to address privacy concerns. In: Senior, A. (ed.) Protecting Privacy in Video Surveillance, pp. 65–89. Springer, Heidelberg (2009)
10. Templeman, R., Korayem, M., Crandall, D., Kapadia, A.: Placeavoider: steering first-person cameras away from sensitive spaces. In: Network and Distributed System Security Symposium (NDSS) (2014)
11. Yamada, T., Gohshi, S., Echizen, I.: Use of invisible noise signals to prevent privacy invasion through face recognition from camera images. In: Proceedings of the 20th ACM International Conference on Multimedia, MM 2012, pp. 1315–1316. ACM, New York (2012)
12. Yamada, T., Gohshi, S., Echizen, I.: Privacy visor: method for preventing face image detection by using differences in human and device sensitivity. In: De Decker, B., Dittmann, J., Kraetzer, C., Vielhauer, C. (eds.) CMS 2013. LNCS, vol. 8099, pp. 152–161. Springer, Heidelberg (2013)
13. Yus, R., Pappachan, P., Das, P.K., Mena, E., Joshi, A., Finin, T.: Demo: faceblock: privacy-aware pictures for google glass. In: Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, pp. 366–366. ACM (2014)

Design and Analysis of Shoulder Surfing Resistant PIN Based Authentication Mechanisms on Google Glass

Dhruv Kumar Yadav¹, Beatrice Ionascu², Sai Vamsi Krishna Ongole³,
Aditi Roy^{3()}, and Nasir Memon^{2,3}

¹ Indian Institute of Technology Kanpur, Kanpur, India
dhruvkr@iitk.ac.in

² New York University Abu Dhabi, Abu Dhabi, United Arab Emirates
bi305@nyu.edu

³ Polytechnic School of Engineering, New York University, Brooklyn, NY, USA
{svo214,ar3824,memon}@nyu.edu

Abstract. This paper explores options to the built-in authentication mechanism of the Google Glass which is vulnerable to shoulder surfing attacks. Two simple PIN-based authentication techniques are presented, both of which provide protection against shoulder surfing. The techniques employ two interfaces for entering the PIN, namely, voice (Voice-based PIN) and touchpad (Touch-based PIN). To enter the same PIN, user has the freedom to choose either technique and thereby interface, as per the environment in which authentication is being performed. A user study was conducted with 30 participants to compare the performance of the proposed methods with the built-in technique. The results show that the proposed mechanisms have a significantly better login success rate than the built-in technique. Interestingly, although the average authentication times of the proposed methods are higher than that of the built-in one, the users perceived them as being faster. The results also indicate that the proposed methods have better perceived security and usability than the built-in method. The study reveals that when it comes to authentication on augmented reality devices, there is a need for authentication mechanisms that complement each other as users tend to prefer a different interface in different contexts.

Keywords: Google Glass · PIN · Authentication · Security · Usability

1 Introduction

Wearable computing devices like the Google Glass [1] are becoming increasingly popular in health-care applications [2–4] and there is a promise of growing adoption in many other innovative applications [5,6]. As with any personal computing device, in order to deter theft and protect personal information, there is a need for an authentication mechanism that binds the specific user to the Glass [7].

In order to authenticate the device user, Google Glass offers a touchpad based pattern lock mechanism. The password is a sequence of four touch gestures chosen from a set of ten gestures, thereby providing the same number of possible passwords as a 4 digit PIN. The gesture set consists of tap or swipe gestures using one or two fingers (see Fig. 1). The user interface for password entry is shown in Fig. 2(a), where the three entered gestures are tap, two-finger tap and swipe back.

The key advantage of the built-in technique is that it is universal in the sense that it can be employed in different ambient conditions including noisy and poorly lit environments. A free hand and a sense of touch is all that is needed. However, the technique has some limitations. First, it is susceptible to shoulder surfing, as the sequence of finger movements on the touchpad located on right side of the head is easily visible to others. Second, some of the gestures (specifically the ones involving the hook swipe gesture) require complicated hand movements and are difficult to perform. Finally, the built-in technique does not exploit the full capabilities of this head mounted device where the screen is only visible to the user.

This paper explores the design and usability of two PIN-based authentication mechanisms that are easy to use while providing reasonable security against shoulder surfing unlike the built-in mechanism offered by Google. The techniques are novel in the sense that two different interfaces can be used to enter the PIN, voice and touch, depending on user choice and the external environment. For *Voice-based PIN* (VBP) entry, a user utters a cipher PIN corresponding to the actual PIN. It should be noted that voice is the natural choice of interface for entering the PIN in a wearable headset device like the Google Glass where textual input is provided by voice. Also, the hands-free format of the mechanism makes it ideal for the device. However, the drawbacks of natural language voice commands, specifically noisy background, prevents it from being used in all contexts. To account for such situations, the other interface through which the Google Glass is controlled, i.e. touchpad, can be employed. Thus, we introduce a second technique, namely *Touch-based PIN* (TBP), which complements the VBP. Both of the techniques leverage the private display visible only to the user to provide protection against shoulder surfing. Although, it could be argued that the display of the Glass is not completely private because the screen is visible from the other side. However, the screen is too small for an adversary to read without being strikingly close to the user or using sophisticated camera equipment. The adversary would have to record and post-process the screen (and also the cipher PIN uttered by the user in the case of VBP) to determine the PIN. This is an unrealistic threat for the average Glass user and thus goes beyond the purpose of this paper.

A user study with 30 participants was conducted to evaluate the security, usability, efficiency, and likability of the proposed authentication mechanisms with respect to the built-in authentication technique. From the user's point of view, any authentication mechanism is bound to fail if perceived as complex or time consuming or vulnerable by the user even though it is theoretically

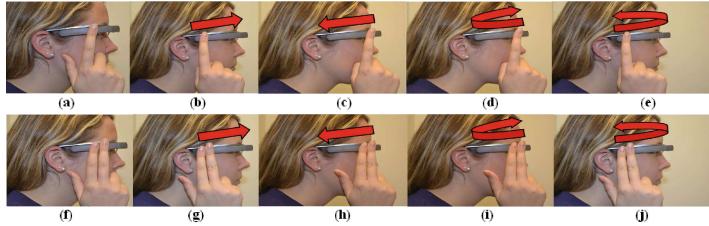


Fig. 1. Available gesture set in the built-in authentication mechanism: (a) tap (b) swipe forward (c) swipe back (d) hook swipe forward (e) hook swipe back (f) two-finger tap (g) two-finger swipe forward (h) two-finger swipe back (i) two-finger hook swipe forward (j) two-finger hook swipe back

resilient. With this in mind it is important to access the perception of security. In this work, experience of the user is investigated and quantified using System Usability Scale (SUS) assessment [27]. To the best of our knowledge, this work is the first to explore alternative authentication mechanisms on the Google Glass with an extensive user study.

The main contributions of the paper are as follows:

- Design of alternate PIN-based password schemes on Google Glass using either voice or touchpad input.
- Quantitative evaluation of efficiency and effectiveness of the PIN-based authentication mechanisms along with the built-in one in terms of login time and login success rate, respectively.
- Qualitative evaluation of the three authentication mechanisms in terms of perceived security, usability, and likelihood of future use.
- Analysis of user reaction and preference in different usage contexts.

Results show that VBP and TBP have a higher perceived security and usability than the built-in method. VBP and built-in methods have comparable login times but TBP requires longer time. However, performance of users improved with number of trials. The study also reveals that users prefer different mechanisms in different contexts. They picked either VBP or TBP over the built-in mechanism in public settings.

The rest of the paper is organized as follows. Section 2 discusses existing authentication mechanisms for various user interfaces. In Sect. 3, the PIN-based authentication schemes are introduced. Section 4 presents the user study design. Evaluation results are presented in Sect. 5. Detailed interpretation of the results is done in Sects. 6 and 7 concludes the paper.

2 Related Work

Personal identification number (PIN) [9] is the most common user authentication method to prevent unauthorized access in modern hand-held computing

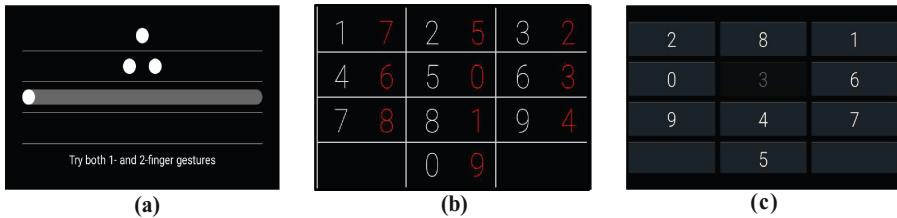


Fig. 2. Layout of the authentication systems. (a) built-in mechanism: the first three entered gestures are tap, two-finger tap, and swipe back (b) VBP mechanism: the white colored digits show the real PIN numbers whereas the red colored digits are the cipher numbers temporarily representing the real PIN digits (c) TBP mechanism: randomly assigned keypad layout that changes with every instance (Color figure online)

devices like mobile phones, tablet, etc. Although in recent times, some devices have incorporated finger-print authentication due to its uniqueness, accuracy, speed and robustness, the mechanism is unlikely to make large impact due to extra cost, additional hardware, non-replaceable nature and fear of identity theft [10]. However, based on short term and lab studies (e.g. [11]), it has been argued that PIN has low memorability and security. To address these issues, multiple methods have been proposed to replace PIN with graphical and pattern-based passwords [12–16]. However, most of them are not robust enough against various vulnerabilities like shoulder surfing and eavesdropping [9]. Several techniques ([17], [18]) were introduced including hardware based [19–21] and biometric feature based ones [22–24] to counter these challenges.

With the evolution of touch-screen based devices, the need for authentication mechanisms better suited for such interfaces became evident. Among the previously mentioned schemes, only pattern-based approaches have seen widespread adoption. The drawmetric scheme Draw-a-Secret [14] is implemented in the Android OS as ‘pattern lock’ [25]. However, one recent study in a real world setting over a reasonable period of time shows that PIN outperforms the pattern lock in terms of speed and error rates [26]. Hence, it can be argued that given the threat model and usability constraints, PIN-based authentication is a reasonable alternative, often in combination with another factor, and will be hard to replace.

The approaches discussed above were designed for devices that have keyboards or touch-screens emulating keyboards. Therefore, none of these methods can be applied in their original form on augmented-reality glasses, like the Google Glass, that do not support traditional user interfaces. Authentication methods that employ the novel interfaces of the Google Glass are yet to emerge. As mentioned before, the built-in mechanism is vulnerable to shoulder surfing attack. Some possible solutions for handling such an attack were outlined in [8]. The idea of voice command based PIN entry mechanism referenced in [8] is similar to the proposed VBP method. The system would display some random number for every digit and the user would have to add his secret digit to this random

number and speak the result mod 10. Manual computation required to enter the PIN makes this method less usable. The other mentioned concept is to display a randomly assigned letter to each digit, and the user has to utter the associated letters corresponding to the PIN digits. However, the authors did not present any prototype design and user study to evaluate the methods. This motivated us to explore alternative authentication schemes for the Google Glass with an aim to increase the security of the built-in method while at least maintaining its current usability level.

3 Authentication Mechanisms

In this section, the authentication mechanisms for voice and touch interfaces are presented in detail.

3.1 Voice-Based PIN (VBP) Authentication

The VBP mechanism uses voice to capture a PIN to unlock the Google Glass. In case the PIN is spoken directly, anyone around the user can hear and learn the secret value. One possible solution is to conceal the real PIN with a cipher text. The ciphers representing the digits could be another set of digits or alphabets (for e.g., $\{A, \dots, Z\}$) or short words (for e.g., $\{\text{'cat'}, \dots, \text{'dog'}\}$). One initial study with different types of ciphers shows that users preferred digits as the cipher in terms of usability and time taken to authenticate. Hence, in our design we map plain text digits to cipher text digits. Of course, the mapping from plain text to cipher text PIN has to change with every instance. For this, the private display visible only to the user can be used.

The Google Glass display is programmed to show a numeric touchpad-like grid, with each cell containing two different digits as shown in Fig. 2(b). The digits on the left in the cells (white ones) represent the real PIN digits found in a standard numeric touchpad. The second digit on the right side (red ones) of each cell shows the randomly mapped cipher digit that the user will have to utter in order to enter the corresponding real PIN digit. So for instance, in Fig. 2(b), cipher digit 7 is used to represent PIN digit 1. To enter the PIN, the user is required to utter the red cipher digits which will then be mapped back to the actual cell numbers to compute the PIN. For instance to enter PIN 3961, a user has to utter 2437 (see Fig. 2(b)).

The voice signal, captured by the mounted microphone on the Google Glass, is then processed by the Android speech recognition API to detect the corresponding cipher digits. Once the digits are recognized, the input PIN is computed by reverse mapping of the cipher digits into the corresponding real PIN numbers. If the detected input PIN matches with the stored PIN, the user is granted access to the Google Glass. It should be noted that the randomly assigned red cipher digits change with every instance.

3.2 Touch-Based PIN (TBP) Authentication

The second authentication mechanism uses the mounted touchpad and requires swipe and tap gestures on the touchpad to enter the PIN. Since the Google Glass has the downward swipe reserved for the ‘go back’ or ‘return to the Home screen’ actions, the proposed method is designed with forward and backward swipes only. Figure 2(c) shows the layout of the user interface for the TBP mechanism. As can be observed from the figure, the cells of the virtual grid are overlaid with different digits which are randomly arranged for each authentication input.

To enter a PIN, the user is required to navigate to each of the corresponding cells using swipes and select the digit by tapping. Forward and backward swipe movements allow the user to move from the current cell to the next cell or previous cell, respectively. To verify a user, the input PIN is compared with the stored one. If they are matched, access is granted.

Since the digit assignment to the grid is different on every instance, the sequence of gestures performed by the user to enter the same PIN will be different each time. Hence, even by observing finger movements on the touchpad, it is difficult to deduce the actual PIN.

4 User Study

A user study was performed to systematically evaluate the properties of the VBP and TBP as alternate authentication mechanisms in terms of their security, usability, effectiveness (e.g., login success rate), efficiency (e.g., login times), and likeability. The study also investigated the following presumptions:

1. The login time for the PIN-based methods will decrease as the user becomes more familiar with the method.
2. The time it takes for authentication and the effort required are negatively correlated with preference.
3. One single mechanism of authentication is not enough for the different environments in which the Google Glass may be used. There is a need for multiple login mechanisms that can be efficiently used in specific situations. A voice based technique such as VBP might be preferred when the user’s hands are busy, whereas the TBP might be suitable in noisy environments or situations where the user is not comfortable speaking the PIN loudly.

We used a Google Glass with 1.2 Ghz dual-core processor, 1 GB RAM, 16 GB memory, and display of 640×360 pixels for the study. The applications were developed on the Android platform 4.4. The user’s identity was protected as no information about the user was stored that would make it possible to identify them.

4.1 Participants

The study was conducted with a total of 30 participants (11 females and 19 males) in a quiet conference room. The subject pool comprised high school,

undergraduate, graduate, PhD students, and faculty without any security expertise. Among the users, only 16.67 % had some prior experience with the Google Glass. 56.67 % of the participants belonged to the 18–24 years age group, 23.33 % to the 25–30 years group, 6.67 % to the 30–35 years group, and the remaining 13.33 % to the 35–40 years group.

75 % of the users reported that they use a locking mechanism for their phone. 70 % of the users strongly agreed that they would be concerned if someone gained access to their Google Glass assuming that the Glass contained or provided access to their personal information such as pictures, videos, messages, and emails. 72.4 % of the users said that they would like to use a locking mechanism for the Glass and that they believe it is important to have it in order to protect their private information.

4.2 User Study Design

The passwords were fixed in advance to limit the number of variables affecting the experiment and to ensure that sufficiently complex passwords were used. Two sets of passwords were chosen as mentioned below:

- Set 1 : PIN was 8340. Pattern was two-finger swipe forward, tap, hook swipe forward, two-finger swipe back.
- Set 2 : PIN was 2791. Pattern was two-finger swipe back, swipe forward, two-finger tap, hook swipe forward.

The PINs were chosen with no repeated, consecutive or neighboring digits. The same PIN was used for the VBP and TBP mechanisms to allow for comparison between the methods. Similarly, for the built-in mechanism, patterns of high strength were chosen. The use of two sets of passwords helped in minimizing the effect of password choice on the analysis.

The group of 30 users was randomly divided into two subgroups and each subgroup was assigned with one of the password sets. Each participant performed three authentication methods in random order to ensure no bias towards a particular mechanism. After an introductory session, each user went through three authentication sessions and a final feedback session as described below.

Introductory Session (5 min): The user was given a short tutorial on how to handle the Google Glass and was allowed to operate and get familiarized with the Glass. The user was assigned the passwords according to the group he/she belonged to.

Authentication Session (approximately 60 min): For each authentication mechanism, the users went through three phases: (1) practice (2) verification, and (3) survey.

At the beginning of each experimental condition, the authentication method was first briefly explained and the user was guided through the authentication interface. The user was allowed to practice the authentication method for a brief time.

Following the practice phase, the user was asked to log into the Google Glass repeatedly until ten successful logins were achieved. The user's inputs, authentication times, and authentication success rates were recorded during this verification phase.

Finally, the user answered a number of questions about usability and security of the authentication mechanism just used before moving on to the next mechanism.

Feedback Session (5 min): At the end of the study, the users were asked a final set of questions comparing all the methods.

4.3 Questionnaire

Part I: At the end of each authentication session, participants were asked to evaluate the security and usability of the method. To get subjective assessment about the usability, the ten-question System Usability Scale (SUS) [27] was used. The term “system” in SUS refers to the “authentication method” to be evaluated in our study. In the SUS assessment, responses to each of the ten questions are given on a five-point scale ranging from “strongly disagree” to “strongly agree”. Apart from the raw SUS score, for better interpretation of the results, SUS percentile [28] and corresponding A-F grading [29] are also reported in this paper.

Part II: Next, five questions were asked about perceived security, convenience, speed, stability, and PIN guessability using the same five-point response scale as follows: (a) I think the method is very secure, (b) I think the method is very convenient, (c) I think the method is very fast, (d) I think the method is very stable, (e) I think the degree of guessability of the PIN is high using this method.

Part III: At the end of the experiment, the users were asked to rank the methods in terms of security, login time, comfort, and likelihood of future use. Some of the questions from Part II were repeated here to study whether the user's perception changes after using all the methods.

Part IV: To get insight about the preference of the users in choosing different mechanisms in different usage situations of the Google Glass, the users were asked to rank the methods based on the likelihood of future usage in the following situations: (a) a quiet classroom or office, (b) a busy subway or a party, (c) at home with family or friends, (d) at home alone, (e) during jogging or biking, (f) listening to loud music alone.

5 Results

This section summarizes the results obtained by the study as described above.

5.1 Login Success Rate

For the VBP, 13 % of the participants accomplished the criterion of 10 successful logins in 10 attempts with no incorrect logins, 63 % participants made 1–3 incorrect logins and the rest required more than 3 attempts making the mean percentage of accuracy for password inputs to be 83 % as shown in Table 1. The reason behind failure to authenticate was one of the following: (a) wrong user input (b) client side error and no speech input due to weak internet connection or heating up of the Google Glass for prolonged use and (c) incomplete voice signal as the user took more time than the allocated one (20 s). It was observed that the utterance of wrong sequence of digits caused error only 32.4 % times while the other two reasons happened 58.1 % and 9.5 % times, respectively.

Table 1. Performance study

	VBP	TBP	Built-in method
Average success rate (%)	83	87	68
Average authentication time (Secs)	6.4	13.9	5.6

For the TBP, 27 % of the users logged in without any error, 57 % accomplished it with maximum 3 incorrect attempts. The overall success rate was 87 %.

For the built-in method, the success rate was lowest, i.e., 68 %. Only 7 % of the users were able to login in the first attempt and 17 % made 1–3 incorrect logins. Most of the users required more than 13 attempts to perform 10 correct logins. Since all the computations were done on the Google Glass itself without information being sent over the network to external server, the only reason behind failure to authenticate for both the TBP and built-in techniques was wrong user input.

5.2 Authentication Time

The authentication time was calculated from the time a participant starts entering the password to the time that he/she successfully logs into the device. It includes the time spent to enter the VBP or TBP (for the built-in method, this is the time to enter the pattern), processing time to perform authentication, and server response delays.

This measure was calculated for the 10 successful trials per participant. Table 1 shows the average time taken by each successful trial for the three authentication mechanisms. It can be observed from these comparisons that the VBP mechanism approximately takes the same authentication time as the built-in method, while the TBP method takes much longer.

The variation in authentication time with 10 login attempts is shown in Fig. 3. The time required for a successful login decreases with successive runs as the users become more skillful and comfortable with the mechanism. The median

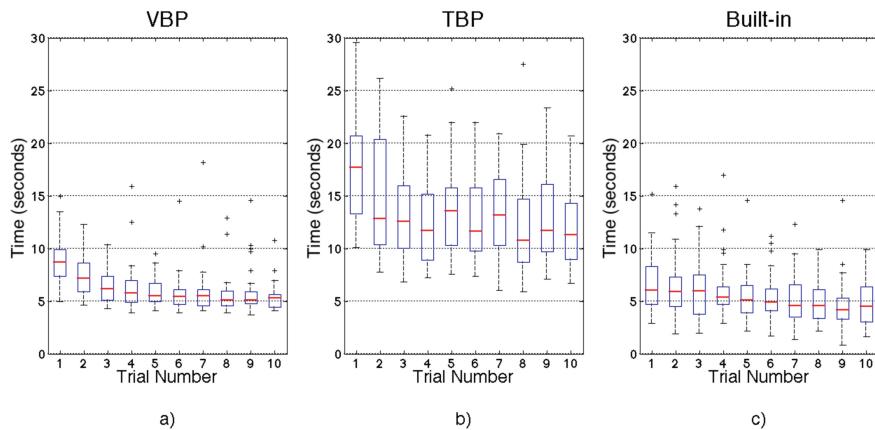


Fig. 3. Variations in login time for 10 trials using (a) VBP (b) TBP (c) built-in method

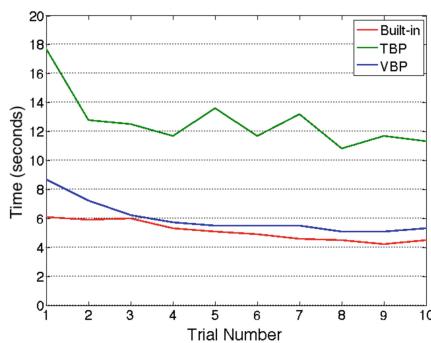


Fig. 4. Variations in median login time over 10 trials using the VBP, TBP and built-in method

Table 2. System Usability Scale (SUS) study

	VBP	TBP	Built-in method
SUS score	76.1 %	69.1 %	61.2 %
SUS response percentile (approx.)	77th	54th	33th
SUS grade	B	C	C

authentication time also indicates a slight learning curve. The learning effect on the speed of authentication is shown in Fig. 4.

5.3 Usability and Security Study

As described in Sect. 4.3 Part I questionnaire, users were asked 10 SUS questions about the respective method. It has been observed from the feedback that

participants rated the VBP high as compared to the other two methods on the usability factors, such as “ease of use” (question 3 of SUS), “would like to use” (question 1 of SUS). Most of them also reported that the VBP could be learned quickly (question 7 of SUS) without taking any technical help (question 4 of SUS) as it was less complex (question 2 of SUS), less cumbersome (question 8 of SUS), and well integrated (question 5 of SUS) compared to the TBP and built-in methods. Users perceived the built-in method to be more complex and hence could not be learned as easily as the two PIN-based methods. Moreover, the built-in method is more inconsistent (question 6 of SUS) and the users had lowest confidence (question 9 of SUS) using the method.

Results of quantitative analysis of the feedback are reported in Table 2. The table shows the average SUS score, response percentile, and grade for each authentication mechanism. The VBP was rated highest with a score of 76.1%, well above the average SUS response value (68%). This marks the VBP in the 77th percentile with a SUS grade of B. The TBP method also scored above the average SUS response value, with score of 69.1%, 54th percentile and a SUS grade of C. The built-in mechanism was rated lowest, with a SUS score of 61.2%, 33th percentile and a SUS score of C.

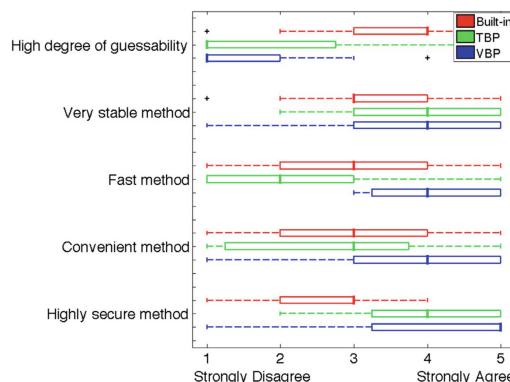


Fig. 5. Overall user experience after using the VBP, TBP, and built-in method

Figure 5 shows the result of Part II questionnaire in Box plot using the same 5-point Likert scale (in the 5 point scale 1 represents “strongly disagree” and 5 represents “strongly agree”). Based on this feedback, the average score for each method is computed as shown in Table 3.

It can be observed from the median ratings in Fig. 5 that the security of the VBP and TBP was thought to be higher than the built-in method. VBP was perceived as most secure with an average score of 4.17 out of 5. Average score for the TBP is 4.14, which is comparable to the VBP. The VBP was also considered to have the lowest degree of PIN guessability with a score of 1.93. The average score of the TBP (2.07) is also significantly lower than that of the built-in method (3.53).

Table 3. Overall user experience results

	VBP	TBP	Built-in method
Security	4.17	4.14	2.50
Convenience	3.53	2.57	3.10
Speed	4.20	2.33	2.90
Stability	3.53	3.90	3.20
Degree of guessability	1.93	2.07	3.53

In terms of convenience, the VBP was rated highest with an average score of 3.53 out of 5, followed by the built-in method and TBP.

The users perceived the VBP to be the fastest one (average score 4.20 out of 5) and the TBP to be slowest (average score 2.33 out of 5). This ranking contradicts with the ranking based on actual average authentication time reported in Table 1, where the built-in method required least time. The perceived speed for the TBP (2.33 in 5) was also close to the built-in method (2.90 in 5). This observation shows that although the VBP or TBP required more processing time, they were not perceived as onerous.

The users also reported that the two PIN-based mechanisms were more stable than the built-in method. The TBP was rated as the most stable method with a score of 3.90.

5.4 Overall User Experience

The feedback data for Part III questionnaire is plotted in Fig. 6.

For security, 53.3 % of the users ranked the VBP first and 46.7 % ranked the TBP first. This result goes along with the feedback taken after each authentication session (discussed in the previous section), which shows that VBP was perceived to be the most secure authentication method.

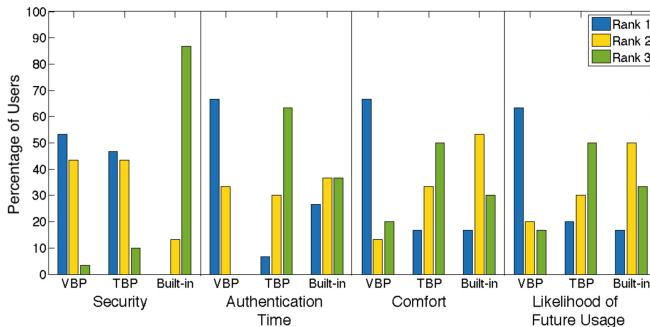


Fig. 6. User rankings of the VBP, TBP, and built-in method in terms of security, authentication time, comfort, and likelihood of future usage

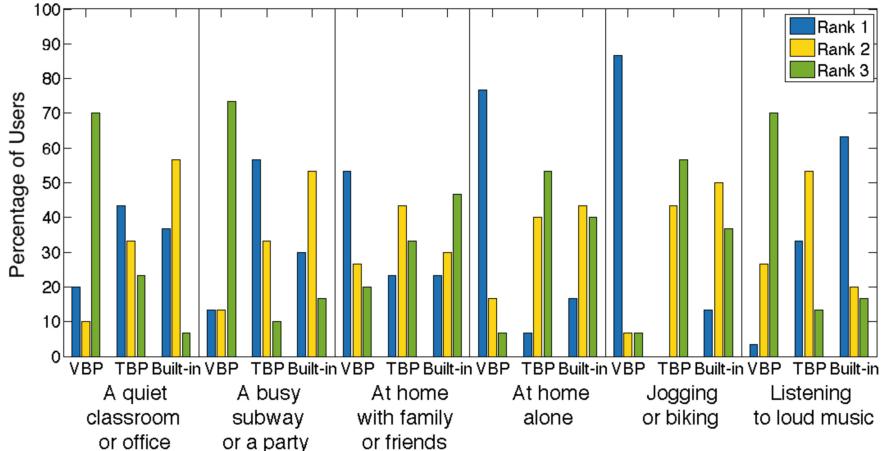


Fig. 7. User rankings of the VBP, TBP, and built-in method in different usage scenarios

In terms of authentication time, 66.7% of the users ranked the VBP first whereas 26.7% ranked the built-in method first. This feedback also supports our previous observation that VBP's perceived speed is higher than that of the built-in method in spite of having longer login time.

In the case of comfort, the VBP was ranked first by 66.7% of the users, whereas the other methods got 16.7% of the votes each. Similarly, 63.3% of the users gave first rank to the VBP for the likelihood of future usage, 20.0% for the TBP, and 16.7% for the built-in mechanism. This shows that the users had a clear inclination towards the PIN-based techniques over the built-in method.

5.5 Context Dependent Preference Study

Figure 7 shows the feedback of the users for Part IV questionnaire.

For the first situation, a quiet office or classroom, 43.3% of the users ranked the TBP as first and 36.7% ranked the built-in method as first. Similarly, for the second situation, a busy subway or social gathering, 56.7% of the users ranked the TBP as first and 30% ranked the built-in method as first. In spite of being slower, users preferred the TBP more than the built-in method in public settings. This shows users' concern about the low security of the built-in method. They were ready to compromise on speed to ensure enhanced security.

For the third situation, at home with family or friends, the VBP was ranked first by 53.3% of the users, whereas the other methods got only 23.3% of the votes each. For the fourth situation, at home alone, 76.7% of the users ranked the VBP as first and only 16.7% ranked the built-in method as first. For the fifth situation, while jogging or biking, 86.7% ranked the VBP as first. Thus, whenever users had a chance to use the VBP (in low background noise situation), they always chose it over any other method.

For the sixth situation, listening to loud music alone, 63.3 % ranked the built-in method as first and 33.3 % ranked the TBP as first. Since the users would be alone, there was less concern about security. Hence, they felt comfortable to use the built-in method in place of the TBP.

The above study shows that while choosing one mechanism, security is the most important criterion to the users. As a consequence, PIN-based mechanisms were preferred in all the public settings. Even if the users were alone, they opted for the VBP if viable. Only in the last case, the built-in method was chosen as security was not a concern and the VBP could not be used.

6 Discussion

Effectiveness: The study provides an understanding of the relative user effort required by the different authentication techniques. With minimal instructions and very little practice, the users were able to login using the TBP and VBP successfully more than 80 % of the time whereas using the built-in mechanism they succeeded only 68 % of the time. This shows that the PIN-based mechanisms are more effective than the built-in one. Moreover, incase of VBP, authentication failure due to incorrect user input accounted for only 32.4 % of all the failed authentication attempts and the rest were due to limitations at the level of Google Glass. With this observation, it is safe to say that VBP will be much more effective with the advancements in the Glass technology.

Efficiency: While the VBP and built-in methods had comparable login times, the TBP needed a significantly longer login time. Multiple horizontal swipes needed to select the desired digits of the PIN may lead to longer login time. On an average, the VBP and built-in method had an 8.0-8.4s shorter login time than the TBP.

However, the actual authentication time does not always match with the users' perception of speed of an authentication mechanism. Although the built-in method outperformed the TBP and VBP in terms of authentication time, participants rated VBP as the fastest. Further, the TBP scored quite close to the built-in mechanism in spite of taking almost double average login time. The effort required to enter a PIN may have affected users' perception of speed.

Learning Effect: One way the PIN input can be speeded up is for users to become more skillful. In our study, all the participants were novice users who were able to increase their speed moderately in the VBP and built-in mechanism over ten trials. However, significant improvement in speed with the TBP was observed, where average login time was halved in 10th trial from the 1st one. This gives us reason to hope that with practice the TBP would be more efficient and acceptable. Thus, the first presumption mentioned in Sect. 4 is validated. Further field studies in natural environments with more experienced users are needed to get a more complete understanding of the learning effect.

Usability: User responses to the SUS were above average for the two PIN-based authentication mechanisms. The built-in mechanism scored lower than average, which indicates that it was not well accepted by the users in spite of being

faster. Overall, users were more satisfied with the new mechanisms compared to the built-in one. This observation was supported by the outcome of the feedback question on likelihood of future usage at the end of the experiment, where participants rated the VBP first followed by the TBP and built-in mechanisms.

Discussion on login time, perceived speed, and likelihood of future usage shows that users always preferred the method with least effort, i.e., VBP. This may not lead to choosing the method with least authentication time. Perceived speed played a more important role in the selection. Thus, our second presumption is partially confirmed.

Security: With respect to security, participants always rated the VBP and TBP much higher than the built-in method. They also reported that the guessability of the PIN using the built-in method is much higher than the two proposed mechanisms. Inclination towards the VBP and TBP was also evident from the users' choice of authentication technique in public space. They always opted for either one of these two methods over the built-in one.

Preferred Mechanism in Hypothetical Usage Situations: Feedback on the likelihood of usage of the three authentication techniques in different usage contexts shows that people tend to choose different authentication mechanisms depending on the environment. Apart from the situational constraints, security was an important aspect in selecting the method. The built-in mechanism was never chosen in public settings.

To choose the preferred authentication method, just like the main menu of the Google Glass, user needs to simply utter 'yes' or 'no' for an authentication option seen on the screen or tap to choose an option. This would allow for a successful integration of the two methods in a real system, without significantly increasing authentication time.

Limitations: The main drawback of the proposed method is that it restricts the use of PINs with repeated digits. The problem is more prominent with VBP, whereas TBP suffers only if there is consecutive occurrences of same digit. Even if a single digit of a PIN is repeated, then the number of guessable passwords reduces to 1000. However, the problem can be solved by changing the cipher digit assignment or the keypad layout after each digit entry for VBP and TBP, respectively.

7 Conclusions

This paper presents two PIN based authentication methods, namely, TBP and VBP for the Google Glass. Both of them take advantage of the unique characteristics of the wearable device, especially the private display, to make the method robust against shoulder surfing and eavesdropping. Since the same PIN can be entered using both methods, a user has the freedom to choose any one depending on the environment.

A detailed user study was conducted to compare the PIN based authentication techniques with the built-in method, testing authentication accuracy,

security, usability and overall user experience. The results show that VBP and TBP have better accuracy than the built-in method, users being able to successfully login more than 80 % of the time using the VBP or TVP, while only 68 % of the time using the built-in mechanism. The VBP and built-in methods have comparable login times (~8 s) but the TBP requires a significantly longer login time (14 s). However, users perceived VBP to be the fastest followed by the built-in and TBP methods. In terms of perceived security, the VBP and TBP were ranked ahead of the built-in method. 53.3 % of the users ranked the VBP and 46.7 % ranked the TBP as the most secure mechanism. The usability of the PIN based methods was also higher as the SUS score for the VBP (76.1 %) and TBP (69.1 %) was better than that for the built-in method (61.2 %). The results also suggest that the PIN based authentication mechanisms have overall better user perception in terms of stability, and likeability than the built-in method. Further, the study provides insight into users' preference when using these three techniques under different contexts. The PIN based techniques complement one another in different scenarios and were preferred more in public settings than the built-in one. The results show that the PIN based methods have the potential to be used for secure user authentication on Google Glass in various usage contexts.

Acknowledgment. This work is supported by the National Science Foundation under Grant No. 1228842.

References

1. Google Glass. <http://www.google.com/glass/start/>
2. Glauser, W.: Doctors among early adopters of google glass. *Can. Med. Assoc. J.* **185**(16), 1385 (2013). doi:[10.1503/cmaj.109-4607](https://doi.org/10.1503/cmaj.109-4607)
3. McNamee, R., Vines, J., Roggen, D., Balaam, M., Zhang, P., Poliakov, I., Olivier, P.: Exploring the acceptability of google glass as an everyday assistive device for people with parkinsons. In: *Proceedings of CHI*, pp. 2551–2554 (2014)
4. Hernandez, J., Li, Y., Rehg, J. M., Picard, R. W.: BioGlass: physiological parameter estimation using a head-mounted wearable device. Accepted in *Mobihealth*
5. Ishimaru, S., Kunze, K., Kise, K., Weppner, J., Dengel, A., Lukowicz, P., Bulling, A.: In the blink of an eye: combining head motion and eye blink frequency for activity recognition with Google Glass. In: *Proceedings of the Augmented Human International Conference*, vol. 15 (2014)
6. Yus, R., Pappachan, P., Das, P. K., Mena, E., Joshi, A., Finin, T.: Demo: Face-Block: privacy-aware pictures for google glass. In: *Proceedings of International Conference on Mobile Systems, Applications, and Services*, vol. 366 (2014)
7. Egelman, S., Jain, S., Portnoff, R. S., Liao, K., Consolvo, S., Wagner, D.: Are you ready to lock? understanding user motivations for smartphone locking behaviors. In: *Proceedings of ACM SIGSAC Conference on Computer and Communications Security* (2014)
8. Bailey, D. V., Drmuth, M., Paar, C.: “Typing” passwords withvoice recognition: how to authenticate to google glass. In: *Proceedings ofthe Symposium on Usable Privacy and Security* (2014)

9. Rogers, J.: Please enter your four-digit pin. In: Financial Services Technology, U.S. Edition, vol. 4 (2007)
10. Ratha, N.K., Chikkerur, S., Connell, J.H., Bolle, R.M.: Generating cancelable fingerprint templates. *IEEE Trans. PAMI* **29**(4), 561–572 (2007)
11. Weiss, R., De Luca, A.: PassShapes: utilizing stroke based authentication to increase password memorability. In: Proceedings of NordiCHI, pp. 383–392 (2008)
12. Davis, D., Monrose, F., Reiter, M.K.: On user choice in graphical password schemes. In: Proceedings of USENIX Security Symposium, vol. 13, pp. 1–14 (2004)
13. Birget, J.-C., Dawei, H., Memon, N.: Graphical passwords based on robust discretization. *IEEE Trans. Inf. Forensics Secur.* **1**(3), 395–399 (2006)
14. Jermyn, I., Mayer, A., Monrose, F., Reiter, M.K., Rubin, A.D.: The design and analysis of graphical passwords. In: Proceedings of SSYM (1999)
15. Dirik, A.E., Memon, N., Birget, J.C.: Modeling user choice in the PassPoints graphical password scheme. In: Proceedings of Usable Privacy and Security, pp. 20–28 (2007)
16. <http://www.passfaces.com/pfphelp/logon.htm>
17. Roth, V., Richter, K., Freidinger, R.: A pin-entry methodresilient against shoulder surfing. In: Proceedings of Conference on Computer and Communications Security, pp. 236–245 (2004)
18. Wiedenbeck, S., Waters, J., Sobrado, L., Birget, J.-C.: Design andevaluation of a shoulder-surfing resistant graphical passwordscheme. In: Proceedings of Conference on Advanced Visual Interfaces, pp. 177–184(2006)
19. Bianchi, A., Oakley, I., Kwon, D.S.: The secure haptic keypad: atactile password system. In: Proceedings of International Conference on Human Factors in Computing Systems, pp. 1089–1092 (2010)
20. Kim, D., Dunphy, P., Briggs, P., Hook, J., Nicholson, J., Nicholson, J., Olivier, P.: Multi-touch authentication ontabletops. In: Proceedings of International Conference on Human Factors in Computing Systems, pp. 1093–1102 (2010)
21. De Luca, A., von Zezschwitz, E., Hussmann, H.: Vibrapass: secureauthentication based on shared lies. In: Proceedings of International Conference on Human Factors in Computing Systems, pp. 913–916 (2009)
22. De Luca, A., Hang, A., Brudy, F., Lindner, C., Hussmann, H.: Touchme once and i know it's you! Implicit authentication based ontouch screen patterns. In: Proceedings of International Conference on Human Factors in Computing Systems (2012)
23. Sae-Bae, N., Memon, N.: Online signature verification on mobiledevices. *IEEE Trans. Inf. Forensics Secur.* **9**(6), 947 (2014)
24. Sae-Bae, N., Memon, N., Isbister, K., Ahmed, K.: Multitouch gesture-based authentication. *IEEE Trans. Inf. Forensics Secur.* **9**(4), 568–582 (2014)
25. Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Dolev, S., Glezer, C.: Google android: a comprehensive security assessment. *Secur. Priv. IEEE* **8**(2), 35–44 (2010)
26. Von Zezschwitz, E., Dunphy, P., De Luca, A.: Patterns in the wild: a field study of the usability of pattern and pin-based authentication on mobile devices. In: Proceedings of International Conference on Human-computer Interaction with Mobile Devices and Services, pp. 261–270 (2013)
27. Brooke, J.: SUS: a quick and dirty usability scale, pp. 189–194. Taylor and Francis (1996)
28. Sauro, J.: Measuring usability with the System Usability Scale (SUS) (2011)
29. Bangor, A., Kortum, P.T., Miller, J.T.: An empirical evaluation of the system usability scale. *Int. J. Hum. Comput. Interact.* **24**(6), 574–594 (2008)

Glass OTP: Secure and Convenient User Authentication on Google Glass

Pan Chan^(✉), Tzipora Halevi, and Nasir Memon

NYU School of Engineering, New York, USA
`{pcl260, thalevi, memon}@nyu.edu`

Abstract. Wearable computing devices have become increasingly popular and while these devices promise to improve our lives, they come with new challenges. This paper focuses on user authentication mechanisms for the Google Glass device (Glass). Glass only has three sources of input: a camera, a microphone, and a touchpad. This limited set of interfaces makes the use of standard passwords infeasible or cumbersome. We therefore propose a One-Time-Password (OTP) authentication scheme, Glass OTP that uses the Glass camera to scan a QR code displayed on the user's smartphone. We implement a proof of concept Glass lock screen which unlocks only upon scanning an OTP generated by the companion Android smartphone application. We also discuss the reliability, security, and convenience of the proposed solution as compared to the current solutions in use.

Keywords: Authentication · User authentication · Wearable computing · Google glass · One time passwords · Password alternative

1 Introduction

As technology advances and miniaturizes, companies have begun to innovate by creating new wearable computing devices ranging from simple bracelets for tracking daily activity [1] to reality augmenting head-mounted displays. The focus of this paper is on the latter type of wearable device, specifically the Glass by Google [2].

Glass presents new challenges in terms of security. Being a device worn on a user's head, Glass is potentially an easier target for thieves. In addition, the limited input capabilities available make the creation of an authentication mechanism that balances security with usability much more difficult. This paper proposes Glass OTP, a simple authentication mechanism for the Glass and discusses its security challenges and considerations of possible solutions.

As of the time of this paper's writing, two authentication applications exist for the Glass. The first and oldest is "Bulletproof" released in April 2013 [7]. The second and most recent one is the lock screen developed by Google and included as of version XE12 [8]. Both of these authentication mechanisms are similar in that they interpret a user's gestures on the touchpad of the Glass to unlock the device.

Bulletproof is a relatively simple lock screen, developed and released as the first lock screen for Glass. It allows a user to configure a combination of single tap, long tap, fling left and fling right to unlock the device. Fling left is a swipe from the back of a user’s head to the front, fling right is the opposite. The combination chosen by a user is written in the source code before the app is built; as a result there is no limit on how long a combination can be [7].

The official Glass lock screen was released and came packaged with the XE12 update in December 2013. This lock screen works similarly to Bulletproof, but has more gesture options and is a more polished product. Available gestures include swiping forwards or backwards with one or two fingers, hook swipes – swiping forward and back in the same motion, and tapping with one or two fingers. The configuration for this lock screen is built into the settings UI of Glass. A user is required to use a combination of exactly four gestures. In the case of a forgotten combination, a user may log in to the MyGlass website and obtain a QR code to reset the Glass lock screen [8].

The touchpad based method of authentication described above aims to be intuitive and easy, but has some limitations in terms of security. In Google’s implementation, there are 4096 possible lock combinations and also a lock out after enough failed attempts. Bulletproof does not have a lock out threshold, but it does allow for a technically unlimited combination length. As a result, both of these implementations appear to mitigate random brute force attacks.

However, there is one major issue that can make these implementations insecure. The unlock pattern combination must be entered by a user on the touchpad which is located by the user’s right temple. This means that the gesture a user enters is very visible to all surrounding people. Unlike traditional “shoulder surfing” attacks, an attacker observing a Glass user from a far is much less obvious and suspicious. This also means that it is very easy for a person to record a user unlocking their Glass from a distance without arousing suspicion. A user can also be unintentionally recorded by surveillance equipment in the vicinity.

Usability of touchpad gesture based unlock patterns on the glass is also of limited value. In a recent study that proposed a novel touch-based pin (TBP) authentication [22], 27 % of the participants achieved successful login without any errors in 10 trials with a success rate of 87 % overall for the method. The TBP method did surpass the Google Glass Pattern Lock mechanism, for which only 7 % of the participants managed to log-in 10 times without any errors and provided 68 % overall success rate. In that study, users indicated they preferred a voice-based pin authentication over TBP, and that both options were preferred over the Glass Pattern Lock mechanism.

The rest of this paper is organized as follows. In the next section, we explore different alternatives that one may employ to create a user authentication mechanism for Glass. In Sect. 3 we present Glass OTP, a One-Time-Password (OTP) authentication scheme that can be used to secure the Glass. Two proof of concept applications that are developed and described: “Glass OTP Companion” application for Android smartphone devices and “Glass OTP” lock screen for Glass devices. In Sect. 4 we analyze the security and usability of the proposed scheme based on a framework developed by Bonneau et al. [9]. In Sect. 5 we conclude with a discussion and a description of possible future work.

2 Options for User Authentication on Glass

In simpler terms, Glass is a wearable device consisting of a small display, which can project a transparent picture in a person's line of view when worn on the head like a normal glass. It has a microphone, a camera, and a touchpad for input. Wi-Fi and Bluetooth are available for connectivity [2].

Glass is a device worn on the head during use and often worn for extended periods of time [2]. As a result, the Glass is potentially an easier target for theft. It is easy to identify someone who is wearing Glass and easy to steal Glass from the head of an inattentive user. With a higher risk of physical theft, the security of the data it contains becomes even more important and hence the need for user authentication.

If we consider the user input mechanisms available on the Glass, we have limited options for user authentication schemes. The standard keyboard based password or pin based authentication implemented in virtually all computing devices are infeasible to implement here as there is no physical keyboard for input, and it appears to be difficult to implement a usable virtual keyboard with the existing Glass hardware. This requires developers to get creative with the hardware provided.

Using the microphone, a user could speak a specific passphrase to unlock the Glass. This would leverage the existing development on voice recognition for Glass [3]. However, the obvious problem here would be an attacker eavesdropping for the passphrase. A novel solution to this has been recently proposed by Yadav et al. [22]. The proposed algorithm employs a pin substitution that is only visible to the user, therefore overcoming the threat from an eavesdropping attack.

Using the touchpad, a user could perform a pattern of gestures to unlock the Glass. There are two existing implementations based on this idea, one being the current official lock screen developed by Google [7, 8]. In Sect. 4 will briefly analyze these existing solutions and their weaknesses.

Aside from traditional user-input, Bluetooth connectivity is available and may be used to transfer data between Glass and a user's smartphone. The use of Bluetooth as a proximity-based locking mechanism was considered prior to the solution presented in this paper. However, there were a few concerns that made Bluetooth an undesirable choice. Bluetooth is often interpreted as a major cause of battery drain and while Bluetooth 4.0's Low Energy mode greatly reduces power consumption [12], it must also be supported by the user's smartphone. For Android, Bluetooth 4.0 LE was not supported until the release of Android 4.3 (API 18) in June 2013 [13]. Additionally, Bluetooth radios commonly used by mobile devices have a range up to 10 meters [12] and determining the distance between devices connected by Bluetooth is not trivial [14]. This would create additional security related concerns. As a result, while Bluetooth may be a possible solution to the problem stated, it is not the solution explored in this work.

Finally, using the front-facing camera, a user could scan a specific image to unlock the Glass. The solution presented in this paper, Glass OTP, relies on this method of input and will be discussed in detail in the next section.

3 Glass OTP

Glass OTP is a new and novel authentication scheme for the Google Glass device, which is designed to be both secure and convenient. The work presented here includes the development of two applications to support this authentication scheme: the “Glass OTP” lock screen for Glass devices and the “Glass OTP Companion” application for Android smartphones.

Glass OTP relies on a private key which is generated by the Glass OTP Companion application and shared with the Glass device. Using this private key, a time based OTP is generated by the Glass OTP Companion application and embedded in a QR code. The Glass OTP lock screen then uses the Glass camera to scan this QR code and verify the OTP before unlocking the device. Code from the ZXing project [6] was used to handle QR generation and QR scanning in the two Glass OTP applications.

3.1 Private Key and OTP Generation

The Glass OTP authentication scheme relies on the HMAC-SHA-256 algorithm to generate OTPs from random symmetric 256-bit keys. The key itself is generated by a smartphone using the Glass OTP Companion application. This application was developed to generate symmetric keys using the built in `javax.crypto.KeyGenerator` class [5]. Using this function, a 256-bit key is created and stored locally on the smartphone device in Android’s “`MODE_PRIVATE`” [4]. Security of this storage method assumes the Android smartphone is not “rooted” or compromised in any way. As for the security of the algorithm chosen; as of the time of this paper’s writing, SHA-256 has been successfully attacked up to 52 rounds out of 64 and still remains secure [10].

Glass OTP uses a time-based scheme following suggestions in IETF RFC 6238 [11] to generate passwords. The hashing algorithm used in Glass OTP, HMAC-SHA-256, is one of two algorithms suggested for implementation of TOTP schemes by the IETF. Glass OTP uses a 30 s time step based on the local system’s current UNIX time. The OTP is obtained by first generating a hash of the current time step using HMAC-SHA-256 initialized with the shared private key. The resulting hash is then converted from the resultant byte array to a binary integer and modulo to the desired OTP length (6 digits in the case of Glass OTP). This implementation is used by both applications involved in the Glass OTP scheme to generate and verify OTPs.

It is realized that this implementation is not without possible vulnerabilities. Relying on local system UNIX time may allow for a replay attack using old OTPs and maliciously modified local time. Obtaining time from verified NTP servers may be one solution. Using local storage in “`MODE_PRIVATE`” [4] is also not foolproof. More secure means of storage include using the built-in “Android Key Store” or hardware-backed “KeyChain” credential storage, both of which were introduced with Android 4.3 (API 18) [15, 16]. The risks mentioned here were deemed acceptable for GlassOTP in its current stage of development and may be mitigated in the future.

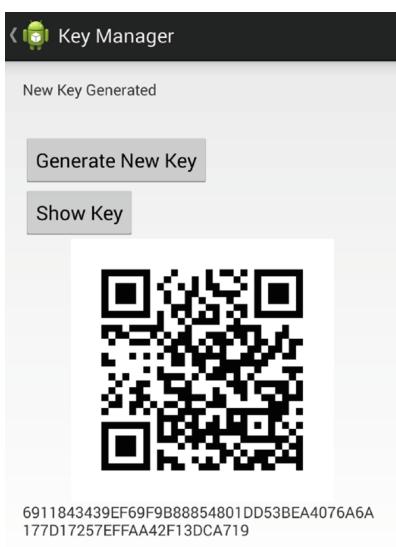


Fig. 1. The key management screen of Glass OTP Companion, where a private key is generated and displayed for scanning.

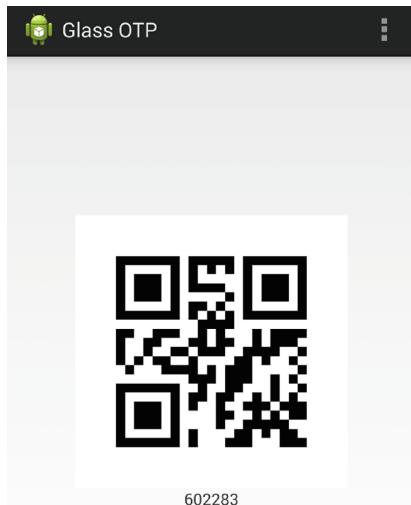


Fig. 2. The main/front screen of Glass OTP Companion where an OTP is generated and displayed for scanning.

3.2 Initial Setup Procedure

The setup procedure for Glass OTP is designed to be quick and requires minimal effort from the user. First, a secret key must be generated by the Glass OTP Companion application on a smartphone. The companion application can then embed the secret key in a QR code using ZXing libraries and display the code for a Glass device running Glass OTP to scan as seen in Fig. 1. Upon scanning the QR code, the Glass OTP lock screen will store the secret key locally and use it to verify scanned OTPs in the future. At this point, setup is complete.

3.3 Unlocking the Glass

Unlocking Glass using Glass OTP is a process designed to be quick and natural, requiring only a glance at one's smartphone. When a user tries to unlock their Glass, Glass OTP will prompt them to scan an OTP QR code. At this point, the user must launch the Glass OTP Companion application on their smartphone. When the companion application launches, it will automatically generate and display an OTP QR code as seen in Fig. 2. Scanning this OTP QR code with the Glass OTP lock screen will unlock the device.

4 Security and Usability Analysis

In a recent paper Bonneau et al. [9] analyzed existing alternatives to traditional passwords. Their work focused on authentication schemes implemented on traditional computing devices (e.g. a PC with keyboard and mouse). As a result, direct comparisons of Glass authentication schemes to analysis done in that research would not be useful. However, Bonneau et al. [9] do outline very specific and detailed criteria upon which to judge an authentication scheme. Below we analyze the proposed Glass OTP authentication scheme against these criteria.

The criteria are separated into three categories: usability, deployability and security. For each criteria, an authentication scheme either “offers the benefit,” “almost offers the benefit,” or “does not offer the benefit.” And for each criteria, an authentication scheme is also evaluated to be “worse than passwords,” “better than passwords,” or “no change.” [9] In our work, we will be comparing Glass OTP against gesture pattern authentication schemes where applicable.

Figure 3 summarizes the analysis against the given criteria and is followed by further explanation of each criterion.

Usability:

- Memorywise-Effortless: The Glass OTP requires nothing to remember. The user must remember to carry his or her phone, but Glass is currently designed as a companion device to smartphones. It can be reasonably assumed the user will carry their smartphone for other reasons as well. This solution is better than pattern locks, where users must memorize a specific sequence of gestures.
- Scalable-for-Users: Each user of Glass OTP is independent from other users and there is no reliance on a support infrastructure, which therefore makes it scalable.
- Nothing-to-Carry: User is required to carry a smartphone to use Glass OTP companion app. Pattern locks have an advantage here, as they do offer this benefit. However, since a large percentage of the population carries a smartphone [20], this does not change the usage model for these users.
- Physically-Effortless: Both OTP and smart phone require some physical effort on the side of the user. For OTP, the user is required to hold a smartphone in front of Glass for scanning, while for pattern locks, physical effort is required to perform gestures correctly. Both lock screens can be set to lock the device only when removed from a user’s head [17].
- Easy-to-Learn: Glass OTP has an easy-to-follow setup procedure and is designed to be simple for the user. This is similar to pattern locks, which also require learning as they are a new authentication scheme.
- Efficient-to-Use: Initial setup is required to exchange and share a private key for the Glass OTP. After initial one-time setup, pointing the Glass at a smartphone to scan a Glass OTP QR code is easy. Some Inconvenience/inefficiency may exist in having to launch the Glass OTP Companion application on the smartphone.

	Glass OTP	Pattern Lock
Usability		
Memorywise-Effortless	●	
Scalable-for-Users	●	●
Nothing-to-Carry		●
Physically-Effortless		
Easy-to-Learn	●	●
Efficient-to-Use		●
Infrequent-Errors	●	
Easy-Recovery-from-Loss	●	●
Deployability		
Accessible		
Negligible-Cost-per-User	●	●
Server-Compatible	N/A	N/A
Browser-Compatible	N/A	N/A
Mature		
Non-Proprietary	●	
Security		
Resilient-to-Physical-Observation	●	
Resilient-to-Targeted-Impersonation	●	
Resilient-to-Throttled-Guessing	●	●
Resilient-to-Unthrottled-Guessing	●	●
Resilient-to-Internal-Observation	●	
Resilient-to-Leaks-from-Other-Verifiers	●	●
Resilient-to-Phishing	●	
Resilient-to-Theft	●	
No-Trusted-Third-Party	●	
Requiring-Explicit-Consent	●	●
Unlinkable	●	●

● = offers the benefit; no circle = does not offer the benefit; N/A = criteria not applicable

Fig. 3. Similar to the table used to compare criteria in “The Quest to Replace Passwords” (Microsoft [9]). This table compares Glass pattern locks and Glass OTP with the given criteria.

- Infrequent-Errors: Glass camera is able to accurately scan the Glass OTP QR code with ease. There are no possible user input errors to account for. This is better than pattern locks, where users must enter a sequence of gestures perfectly or try again.
- Easy-Recovery-from-Loss: A user is able to save their generated private key, either as text or as a screenshot of the private key QR code. Recovery involves scanning this code with a new smartphone running the Glass OTP Companion app. This is similar to the official Glass lock screen’s recovery feature [8].

Deployability:

- Accessible: Glass OTP requires more physical effort due to necessary handling of the smartphone. Although the effort may be negligible, it must still be considered in this analysis. Pattern locks are similar as they require physical effort to lift one's arm to the proper position and perform the correct gestures.
- Negligible-Cost-per-User: Glass is designed as a companion device to smart phones. It is assumed that the Glass user will have a smart phone capable of running the Glass OTP companion app.
- Server-Compatible: Glass OTP functions are entirely client-side, with no requirement for web-hosted resources. Pattern locks are similar in this criteria.
- Browser-Compatible: Glass OTP isn't meant as an everyday password replacement for multiple scenarios.
- Mature: This protocol introduces the first Glass security application that utilizes OTPs. Patterns locks for the Glass are also fairly immature, having existed only as early as April 2013 [7].
- Non-Proprietary: Glass OTP will be open source once released and would be non-proprietary. Google's pattern lock is proprietary [8], Bulletproof pattern lock is open source [7].

Security:

- Resilient-to-Physical-Observation: Any observed OTP QR code is time sensitive and useless after a short period of time, and is therefore resilient to physical observations. This offers better solution than pattern locks, where a user's gestures are clearly visible to nearby observers.
- Resilient-to-Targeted-Impersonation: Users decide how they want to keep their own private keys securely backed up. Better than pattern lock, as Google's pattern lock can be reset if an attacker obtains access to a victim's Google account [8].
- Resilient-to-Throttled-Guessing: It is infeasible to try all possible number combinations for the OTP before the current valid combination would expire, which renders it resilient to throttled guessing. Google's pattern lock has a lockout threshold [8]. Bulletproof has no limit on pattern length [7].
- Resilient-to-Unthrottled-Guessing: It is infeasible to try all possible number combinations before the current valid combination would expire. Google's pattern lock has a lockout threshold [8]. Bulletproof has no limit on pattern length [7].
- Resilient-to-Internal-Observation: An intercepted OTP would expire before it is useful. It is infeasible that an attacker could intercept an OTP, steal the user's Glass and authenticate before the OTP expires. This assumes the user does not store private key in an insecure manner. I.e. user does not store private key on Glass or smartphone in unencrypted form. The mechanism is better than pattern locks, where if a user's touchpad input is intercepted; the pattern of gestures for unlocking the device can be obtained and replayed.
- Resilient-to-Leaks-from-Other-Verifiers: All verification is done locally by Glass OTP, which renders it resilient to such leaks. Pattern locks also perform verification locally.

- Resilient-to-Phishing: All verification is done locally by Glass OTP. This is better than pattern locks, where a successful phishing attack on a victim’s Google account would grant access to the Glass device if using pattern lock [8].
- Resilient-to-Theft: Compromise of Glass OTP requires theft of Glass device and paired smartphone. Resilience to theft depends on the security of the user’s smartphone. This is better than pattern lock that does not rely on a separate physical device for authentication (only requires theft of Glass and pattern knowledge).
- No-Trusted-Third-Party: Glass OTP only allows a user to keep his own private key locally. All verification is done locally by Glass OTP. This is better than pattern lock which relies on Google services for pattern recovery.
- Requiring-Explicit-Consent: Glass OTP requires a user to purposefully use and position their phone in front of Glass.
- Unlinkable: All verification is done locally by Glass OTP. Same as pattern locks, which also verify locally.

Glass Camera Accuracy and Reliability. Another concern regarding the Glass OTP authentication scheme is the reliance on the Glass’s front facing camera. Questions have been raised about if the camera can quickly and accurately scan QR codes displayed by the smartphone in various conditions. The distance a smartphone must be from the Glass camera is also something to be considered. Tests performed until the time of this paper’s writing has yielded good results, however these tests were not done under properly controlled environments so official results were not recorded. The following results are recollections of experience from hands-on testing in every-day real world environments.

The Glass camera and ZXing code [6] (for QR code interpretation) is able to quickly and easily scan QR codes under most conditions. In normal light, such as outdoors in overcast conditions or indoors with regular lighting, QR codes are quickly and easily scanned. In low light or no light, the same was true since the QR code is being displayed by a smartphone screen which is backlit. In these cases, the smartphone displaying a QR code could be held away as far as one to two feet (almost half an average arm’s length) away from Glass and still be successfully scanned.

The only issues arose during testing in bright sunlight. In some cases, direct sunlight was able to overpower the backlighting of smartphone screens and make scanning the displayed QR code difficult to impossible. Indirect sunlight also posed an issue with smartphones which had insignificant maximum screen brightness. On newer phones, such as the HTC One M8 used for testing, max brightness was enough to display a clear QR code for the Glass to scan in bright but indirect sunlight. In the worst cases, the QR code on a smartphone could still be scanned if the user shaded the phone with his or her hand; bringing the phone closer to the camera also improved chances of a successful scan.

Key Backup and Recovery. A major concern with any kind of authentication scheme, especially ones which rely on a second physical token, is how to back-up and restore a key. With Glass OTP, the solution is fairly simple but places more responsibility on the user. Currently, a user can have the Glass OTP Companion application

display the saved private key as both a string and QR code at any time. The easiest way to back up the private key would be to take a screenshot of the QR code representation and store it somewhere safe. If the smartphone used to unlock a specific Glass device is ever lost, a user just has to scan the private key QR code with the Glass OTP Companion application installed on a new smartphone.

5 Conclusions and Future Work

Wearable devices are improving and becoming increasingly popular as time goes on in our electronically connected society. Some people have a desire for as much information as possible, as fast as possible, and as accessible as possible. Wearables promise to fulfill such desires, but in doing so come with large privacy and security concerns.

Google Glass is such a device; it aims to keep a user connected at all times, containing and providing data relevant, specific, and sometimes private to the user. Unfortunately, as a result of its form factor, passwords as we know it cannot be used to secure the device. There is no keyboard to input long, random, and secure passwords.

Existing solutions aim to provide an alternative relying on gestures on the touchpad. But these gesture combinations are out in the open, for anyone nearby to see and worse, record.

This paper presents Glass OTP, an authentication scheme which finds a balance security and convenience not seen in existing authentication schemes. Glass OTP has offers resiliency to numerous security risks, similar to traditional One-Time-Password password schemes [9]. However, Glass OTP overcomes the inconvenience associated with One-Time-Passwords by using the Glass camera, sparing the user from having to manually input the password before it becomes invalid.

Future solutions could be introduced alongside hardware improvements by Google. For example, adding a biometric security mechanism similar to Apple's Touch ID [18] could prove to be a good solution compared to the alternatives examined in this paper. With the rising affordability of fingerprint readers, and as new applications –such as the Paypal app [19] – begin accepting fingerprints for authentication, integrating fingerprint readers into Google Glass may offer a promising authentication solution. However, this requires changes to the hardware that are not currently available on the system.

Future work should include a user study that examines the security and the usability of the OTP mechanism vs. the Pattern Lock mechanism. It should compare the actual time and the perceived time and convenience of using each approach. As users previously expressed significant concerns about privacy issues posed by Google Glass [21], the effect of adding a locking mechanism should be explored, and whether it will raise potential users' willingness to accept the system.

Acknowledgments. This work was supported in part by the NSF (under grant 1228842). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

References

1. Fitbit, Inc. Fitbit Official Site. <http://www.fitbit.com>. Accessed 2 October 2014
2. Google. Google Glass. <http://www.google.com/glass>. Accessed 20 September 2014
3. Google. Google Developers Voice Input. <https://developers.google.com/glass/develop/gdk/voice>. Accessed 8 October 2014
4. Google. Context Android Developers. <http://developer.android.com/reference/android/content/Context.html>. Accessed 1 October 2014
5. Oracle. KeyGenerator (Java Platform SE 7). <http://docs.oracle.com/javase/7/docs/api/javax/crypto/KeyGenerator.html>. Accessed 1 October 2014
6. Github, Inc. zxing/Zxing. <https://github.com/zxing/zxing>. Accessed 12 October 2014
7. Github, Inc. kaze0/Bulletproof. <https://github.com/kaze0/bulletproof>. Accessed 10 October 2014
8. Google. Screen Lock – Google Glass Help. <https://support.google.com/glass/answer/4389349?hl=en>. Accessed 8 October 2014
9. Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. Microsoft. <http://research.microsoft.com/pubs/161585/QuestToReplacePasswords.pdf>
10. Li, J., Isobe, T., Shibusaki, K.: Converting MITM Preimage Attack into Pseudo Collision Attack: Application to SHA-2 (2012). Sony China Research Laboratory. Sony Corporation. <http://fse2012.inria.fr/SLIDES/67.pdf>
11. M'Raihi, D., Machani, S., Pei, M., Rydell, J.: TOTP: Time-Based One-Time Password Algorithm. Int. Eng. Task Force (2011). <https://tools.ietf.org/html/rfc6238>
12. Bluetooth SIG, Inc. Basics | Bluetooth Technology Website. <http://www.bluetooth.com/Pages/Basics.aspx>. Accessed 10 December 2014
13. Google. Bluetooth Low Energy | Android Developers. <https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>. Accessed 12 December 2014
14. Ghose, A., Bhaumik, C., Chakravarty, T.: BlueEye – A System for Proximity Detection Using Bluetooth on Mobile Phones. UbiComp. <http://www.ubicomp.org/ubicomp2013/adjunct/adjunct/p1135.pdf>
15. Google. Android 4.3 APIs | Android Developers. <https://developer.android.com/about/versions/android-4.3.html>. Accessed 16 December 2014
16. Elenkov, N.: Jelly Bean hardware-backed credential storage. <http://nelenkov.blogspot.com/2012/07/jelly-bean-hardware-backed-credential.html>. Accessed 16 December 2014
17. Google. On-Head Detection – Google Glass Help. <https://support.google.com/glass/answer/3079857>. Accessed 20 December 2014
18. Apple. Apple – iPhone 6 – Touch ID. <https://www.apple.com/iphone-6/touch-id/>. Accessed 20 December 2014
19. FinExtra. PayPal adds fingerprint authentication to more Samsung devices (2014). <http://www.finextra.com/news/announcement.aspx?pressreleaseid=56577&topic=retail>
20. PewResearch Internet Project, Mobile Technology Fact Sheet. <http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/>
21. Cnet, 72 percent say no to Google Glass because of privacy. <http://www.cnet.com/news/72-percent-say-no-to-google-glass-because-of-privacy/>
22. Yadav, D.K., Ionascu, B., Ongole, S.V.K., Roy, A., Memon, N.: Design and analysis of shoulder surfing resistant PIN based authentication mechanisms on google glass. In: Wearable S&P 2015 (2015)

Author Index

- Ali, Syed Taha 34
Andrychowicz, Marcin 1
- Böhme, Rainer 19
- Caine, Kelly 231
Chan, Pan 298
Cheon, Jung Hee 142, 194
- Dabrowski, Adrian 274
Dai, Wei 160
Doröz, Yarkin 160
Dziembowski, Stefan 1
- Gaucas, Dale 78
Grossklags, Jens 63
- Halevi, Tzipora 298
Hao, Feng 34
Hell, Martin 261
Hileman, Garrick 92
- Ionascu, Beatrice 281
- Jacques, Robert St 78
Johansson, Bjorn 261
Johnson, Benjamin 63
- Kim, Miran 142, 194
Kim, Myungsun 142
Kohno, Tadayoshi 94
Krishna Ongole, Sai Vamsi 281
Krombholz, Katharina 274
- Lantz, Patrik 261
Laszka, Aron 63
Lauter, Kristin 194
Lee, Peter Hyun-Jeen 34
- Lepoint, Tancrede 184
Lerner, Adam 94
- Malinowski, Daniel 1
Mazurek, Lukasz 1
McCorry, Patrick 34
McReynolds, Emily 94
Meiklejohn, Sarah 127
Memon, Nasir 281, 298
Möser, Malte 19
Motti, Vivian Genaro 231
- Orlandi, Claudio 127
- Pullonen, Pille 172
- Roesner, Franziska 94
Rowan, Brendan 112
Roy, Aditi 281
- Scott, Will 94
Siim, Sander 172
Smeets, Ben 261
Smith, Matthew 274
Sunar, Berk 160
- Tibouchi, Mehdi 184
Toorani, Mohsen 245
Tromp, John 49
- Valenta, Luke 112
Vandervort, David 78
Varia, Mayank 213
- Weippl, Edgar 274
- Yadav, Dhruv Kumar 281
Yakoubov, Sophia 213
Yang, Yang 213