Shai Halevi (Ed.)

Advances in Cryptology – CRYPTO 2009

29th Annual International Cryptology Conference
Santa Barbara, CA, USA, August 16-20, 2009
Proceedings

Springer

Volume Editor

Shai Halevi
IBM Research
Hawthorne, NY, USA
E-mail: shaih@alum.mit.edu

Library of Congress Control Number: 2009931222

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, I.1

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-642-03355-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-03355-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12725359 06/3180 5 4 3 2 1 0

Preface

CRYPTO 2009, the 29th Annual International Cryptology Conference, was sponsored by the International Association for Cryptologic Research (IACR) in cooperation with the IEEE Computer Society Technical Committee on Security and Privacy and the Computer Science Department of the University of California at Santa Barbara. The conference was held in Santa Barbara, California, during August 16–20, 2009, and John Black served as the General Chair. The Program Committee consisted of 29 members and two advisory members, whose names are listed on the next page, and I had the privilege of serving as the Program Chair.

The conference received 213 submissions. The Program Committee, aided by 217 external reviewers, reviewed all these submissions and discussed them in depth. After an intensive review period of 11 weeks, the committee accepted 40 of these submissions. Two pairs of submissions were merged, yielding a total of 38 papers in the technical program of the conference. These proceedings include the revised versions of the 38 papers that were presented at the conference. These revised papers were not subject to editorial review and the authors bear full responsibility for their contents. The best-paper award was given to the paper "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate" by Stevens, Sotirov, Appelbaum, Lenstra, Molnar, Osvik, and de Weger.

The conference featured two invited lectures: one by Ed Felten and the other by Ueli Maurer. An abstract of Maurer's talk, titled "Abstraction in Cryptography," is included in these proceedings. The program also included a Rump Session, featuring short informal talks on recent results and work in progress.

I wish to thank all the authors who submitted their work to CRYPTO 2009. We received a large number of high-quality submissions, and even though we accepted more submissions than usual, there were still many good ones that we just could not fit in the program (but surely they will be published elsewhere). I am proud to be working in a field that consistently produces such strong results.

I owe a debt of gratitude to members of the Program Committee for their outstanding work. Evaluating such a large number of submissions in the short review period is very demanding, and the committee members contributed their knowledge and time to ensure that all submissions were reviewed in depth. Many thanks also to all the external reviewers who helped us with this task. I also thank Christof Paar, Christopher Wolf, and Alexander May for their help with organizing the PC meeting. And of course, I am thankful for the support that I received from all the members of the Cryptography group in IBM T.J. Watson Research Center: Rosario Gennaro, Craig Gentry, Charanjit Jutla, Jonathan Katz, Hugo Krawczyk, Tal Rabin, and Vinod Vaikuntanathan.

June 2009 Shai Halevi

CRYPTO 2009

The 29th International Cryptology Conference

August 16–20, 2009, Santa Barbara, California, USA

Sponsored by the
International Association for Cryptologic Research (IACR)

in cooperation with
IEEE Computer Society Technical Committee on Security and Privacy,
Computer Science Department, University of California, Santa Barbara

General Chair

John Black	University of Colorado at Boulder, USA

Program Chair

Shai Halevi	IBM Research, USA

Program Committee

Masayuki Abe	NTT, Japan
Dan Boneh	Stanford University, USA
Christophe De Cannière	Katholieke Universiteit Leuven, Belgium
Jean-Sébastien Coron	University of Luxembourg, Luxembourg
Marc Fischlin	Technische Universität Darmstadt, Germany
Steven Galbraith	Royal Holloway, UK
Shafi Goldwasser	MIT, USA and Weizmann Institute, Israel
Jens Groth	University College London, UK
Iftach Haitner	Microsoft Research, USA
Yuval Ishai	Technion, Israel and UCLA, USA
Marc Joye	Thomson R&D, France
Jonathan Katz	University of Maryland and IBM Research, USA
Kaoru Kurosawa	Ibaraki University, Japan
Anna Lysyanskaya	Brown University, USA
Phong Q. Nguyen	INRIA and ENS, France
Jesper Buus Nielsen	University of Aarhus, Denmark
Christof Paar	Ruhr-Universität Bochum, Germany
Rafael Pass	Cornell University, USA
Chris Peikert	SRI International, USA
Krzysztof Pietrzak	CWI Amsterdam, The Netherlands

Sebastiaan Indesteege
Tetsu Iwata
Stas Jarecki
Antoine Joux
Yael Tauman Kalai
Bhavana Kanukurthi
Markus Kaspar
Stefan Katzenbeisser
Yutaka Kawai
Aggelos Kiayias
Eike Kiltz
Markulf Kohlweiss
Gillat Kol
Yuichi Komano
Takeshi Koshiba
Noboru Kunihiro
Alptekin Küpçü
Anja Lehmann
Reynald Lercier
Gaëtan Leurent
Benoît Libert
Huijia Lin
Yehuda Lindell
Richard Lindner
Moses Liskov
Feng-Hao Liu
Vadim Lyubashevsky
Hiren Maharaj
Mohammad Mahmoody-Ghidary
Alexander May
Catherine Meadows
Alfred Menezes
Daniele Micciancio
Payman Mohassel
Tal Moran
Ciaran Mullan
David Naccache
Moni Naor
Gregory Neven
Gonzaler Nieto
Svetla Nikova
Ryo Nishimaki
Ryo Nojima
Martin Novotny
Koji Nuida
Adam O'Neill
Wakaha Ogata
Miyako Ohkubo
Cristina Onete
Claudio Orlandi
Carles Padro
Pascal Paillier
Omkant Pandey
David Parkes
Jacques Patarin
Olivier Pereira
Ludovic Perret
Christiane Peters
David Pointcheval
Carl Pomerance
Christopher Portmann
Manoj M Prabhakaran
Jörn Müller Quade
Tal Rabin
Dominik Raub
Christian Rechberger
Omer Reingold
Leonid Reyzin
Vincent Rijmen
Matthieu Rivain
Phillip Rogaway
Alon Rosen
Guy Rothblum
Andy Rupp
Amit Sahai
Palash Sarkar
Werner Schindler
Christian Schridde
Dominique Schröder
Michael Scott
Gil Segev
Nicolas Sendrier
abhi shelat
Emily Shen
Amir Shpilka
Thomas Shrimpton
Nigel Smart
François-Xavier Standaert
Cyril Stark
Damien Stehlé
Asgeir Steine
Ron Steinfeld
Marc Stevens
Makoto Sugita
Willy Susilo
Koutarou Suzuki
Bjoern Tackmann
Keisuke Tanaka
Stefano Tessaro
Marco Tomamichel
Nikos Triandopoulos
Wei-lung Tseng
Yasuyuki Tsukada
Jorge Jimenez Urroz
Alexander Ushakov
Berkant Ustaoglu
Vinod Vaikuntanathan
Mayank Varia
Maria Isabel Gonzalez Vasco
Muthu Venkitasubramaniam
Frederik Vercauteren
Damien Vergnaud
Thomas Vidick
Charlotte Vikkelsoe
Ivan Visconti
David Wagner
Shabsi Walfish
Huaxiong Wang
Bogdan Warinschi
Brent Waters
Hoeteck Wee
Enav Weinreb
Susanne Wetzel
Daniel Wichs
Christopher Wolf
Stefan Wolf
Hongjun Wu
David Xiao
Sergey Yekhanin
Hila Zarosim
Yunlei Zhao
Hong-Sheng Zhou
Vassilis Zikas

Table of Contents

Reconstructing RSA Private Keys from Random Key Bits

Nadia Heninger[1] and Hovav Shacham[2]

[1] Princeton University
nadiah@cs.princeton.edu
[2] University of California, San Diego
hovav@cs.ucsd.edu

Abstract. We show that an RSA private key with small public exponent can be efficiently recovered given a 0.27 fraction of its bits at random. An important application of this work is to the "cold boot" attacks of Halderman et al. We make new observations about the structure of RSA keys that allow our algorithm to make use of the redundant information in the typical storage format of an RSA private key. Our algorithm itself is elementary and does not make use of the lattice techniques used in other RSA key reconstruction problems. We give an analysis of the running time behavior of our algorithm that matches the threshold phenomenon observed in our experiments.

1 Introduction

In this paper, we present a new algorithm for the problem of reconstructing RSA private keys given a random δ-fraction of their bits. For RSA keys with small public exponent, our algorithm reconstructs the private key with high probability when $\delta \geq 0.27$. The runtime analysis of our algorithm relies on an assumption (Conjecture 1) and is thus heuristic; but we have verified experimentally that it succeeds with high probability.

Motivation: cold boot attacks. An important application of our algorithm is key recovery from the randomly distributed unidirectional bit corruption observed in the recent work of Halderman et al. [10], which demonstrated that DRAM remanence effects make possible practical, nondestructive attacks that recover (a degraded version of) secret keys stored in a computer's memory. Using these "cold boot" attacks, attackers with physical access to a machine can break popular disk encryption systems or recover an SSL server's private key.

One consequence of the nature of the attack is that a perfect image of the contents of memory may not be available to the attacker; instead, some bits may have been flipped. Halderman et al. observe that, within a DRAM region, the decay is overwhelmingly either $0 \rightarrow 1$ or $1 \rightarrow 0$. The decay direction for a region can be determined by comparing the number of 0s and 1s. (In an uncorrupted key we expect these to be approximately equal.) For a region of $1 \rightarrow 0$ decay, a 1 bit in the decayed version is known (with high probability) to correspond to a

1 bit in the original key, whereas a 0 bit might correspond to either a 0 or 1 bit in the original key. If a ρ fraction of bits decays and 0s and 1s were present in equal numbers in the key then we will know, given the degraded representation, a $\delta = (1 - \rho)/2$ fraction of key bits.

Halderman et al. further showed that it is possible to exploit redundancy in key data to create algorithms for reconstructing DES, AES, and cipher tweak keys from their degraded in-memory representations. In addition, they experimented with reconstructing RSA keys by using the public modulus N to correct its partly-known factors p and q. We extend this idea to take into account other fields of an RSA private key and provide an analysis of the resulting algorithm's runtime behavior. Our improvement makes a significant difference in practice: their algorithm takes several minutes to recover a 2048-bit RSA key from 12% unidirectional corruption; ours takes under a second to recover a 2048-bit key from as much as 46% unidirectional corruption.

Our algorithm and its performance. Our two main results in this paper are: (1) an algorithm for reconstructing RSA private keys given a random δ-fraction of their bits; and (2) an analysis of the algorithm's runtime behavior for random inputs that shows that it will succeed in expected quadratic time when $\delta \geq .27$. The runtime analysis depends crucially on both a uniformly random distribution of known bits and the assumption that the effect of a bit error during reconstruction is propagated uniformly through subsequent bits of the key.

Our algorithm performs better than the algorithm given by Halderman et al. because it is able to make use of five components of the RSA private key: p, q, d, d_p, and d_q. We can use known bits in d, d_p, and d_q to make progress where bits in p and q are not known. To relate d to the rest of the private key, we make use of techniques due to Boneh, Durfee, and Frankel [4]; to relate d_p and d_q to the rest of the private key, we make new observations about the structure of RSA keys that may be of independent interest. This is discussed in Section 2.

If the algorithm has access to fewer components of the RSA private key, the algorithm will still perform well given a sufficiently large fraction of the bits. For example, it can efficiently recover a key given

$\delta = .27$ fraction of the bits of p, q, d, d_p, and d_q.
$\delta = .42$ fraction of the bits of p, q, and d.
$\delta = .57$ fraction of the bits of p and q.

The reconstruction algorithm itself, described in Section 3, is elementary and does not make use of the lattice basis reduction or integer programming techniques that have been applied to other kinds of RSA key reconstruction problems. At each step, it branches to explore all possible keys, and prunes these possibilities using our understanding of the structure of RSA keys and the partial information we are given about key bits. We give an analysis of the algorithm for random inputs in Section 4. We obtain a sharp threshold around $2 - 2^{(4/5)} \approx 27\%$ of known key bits. Below this threshold, the expected number of keys examined is exponential in the number of bits of the key, and above this threshold, the expected number of keys examined is close to linear. Note that this threshold applies only to our particular approach. We suspect these results could be improved using more sophisticated methods.

Finally, we have implemented our algorithm and performed extensive experiments using it. The results are described in Section 5. The algorithm's observed behavior matches our analytically derived bounds and validates the heuristic assumptions made in the analysis.

Small public-exponent RSA. Our algorithm is specialized to the case where *the public exponent e is small.* The small-e case is, for historical reasons, the overwhelmingly common one in deployed RSA applications such as SSL/TLS. For example, until recently Internet Explorer would reject TLS server certificates with an RSA public exponent longer than 32 bits [5, p. 8]. The choice $e = 65537 = 2^{16} + 1$ is especially widespread. Of the certificates observed in the UCSD TLS Corpus [23] (which was obtained by surveying frequently-used TLS servers), 99.5% had $e = 65537$, and all had e at most 32 bits.

Related work. Inspired by cold boot attacks, Akavia, Goldwasser, and Vaikuntanathan [1] formally introduced *memory attacks*, a class of side-channel attacks in which the adversary is leaked a (shrinking) function of the secret key. One research direction, pursued by Akavia, Goldwasser, and Vaikuntanathan and, in followup work, Naor and Segev [18], is constructing cryptosystems provably secure against memory attacks.[1] Another research direction is to evaluate the security of existing cryptosystems against memory attacks. Our work is along this latter direction.

There is a great deal of work on both factoring and reconstructing RSA private keys given a fraction of the bits.

Maurer [14] shows that integers can be factored in polynomial time given oracle access to an ϵ fraction of the bits of a factor.

In a slightly stricter model, the algorithm has access to a fixed subset of consecutive bits of the integer factors or RSA private keys. Rivest and Shamir [21] first solved the problem for a 2/3-fraction of the least significant bits of a factor using integer programming. This was improved to 1/2 of the least or most significant bits of a factor using lattice-reduction techniques pioneered by Coppersmith [6]; we refer the reader surveys by Boneh [3] and May [16] as well as May's Ph. D. thesis [15] for bibliographies. More recently, Herrmann and May extended these techniques to efficiently factor given at most $\log \log N$ known blocks of bits [12].

The problem we seek to solve can be viewed as a further relaxation of the conditions on access to the key bits to a fully random subset. These lattice-reduction techniques are not directly applicable to our problem because they rely on recovering *consecutive* bits of the key (expressed as small integer solutions to modular equations), whereas the missing bits we seek to find are randomly distributed

[1] There has been substantial other recent work on designing cryptosystems secure in related key-leakage models (e.g., [20,8,2]); for a survey, see Goldwasser's invited talk at Eurocrypt 2009 [9] and the references therein.

throughout the degraded keys. It is possible to express our reconstruction problem as a knapsack, and there are lattice techniques for solving knapsack problems (see, e.g., Nguyen and Stern [19]), but we have not managed to improve on our solution by this approach.

2 RSA Private Keys

The PKCS#1 standard specifies [22, Sect. A.1.2] that an RSA private key include at least the following information:

– the (n-bit) modulus N and public exponent e;
– the private exponent d;
– the prime factors p and q of N;
– d modulo $p-1$ and $q-1$, respectively denoted d_p and d_q; and
– the inverse of q modulo p, denoted q_p^{-1}.

In practice, an RSA key in exactly this format can be recovered from the RAM of a machine running Apache with OpenSSL [10]. The first items – N and e – make up the public key and are already known to the attacker. A naïve RSA implementation would use d to perform the private-key operation $c \mapsto c^d \bmod N$, but there is a more efficient approach, used by real-world implementations such as OpenSSL, that is enabled by the remaining private-key entries. In this approach, one computes the answer modulo p and q as $(c \bmod p)^{d_p}$ and $(c \bmod q)^{d_q}$, respectively; then combines these two partial answers by means of q_p^{-1} and the Chinese Remainder Theorem (CRT). This approach requires two exponentiations but of smaller numbers, and is approximately four times as fast as the naïve method [17, p. 613].

Observe that the information included in PKCS#1 private keys is *highly redundant.* In fact, knowledge of any single one of p, q, d, d_p, and d_q is sufficient to reveal the factorization of N.[2] It is this redundancy that we will use in reconstructing a corrupted RSA key.

We now derive relations between p, q, d, d_p, and d_q that will be useful in mounting the attack. The first such relation is obvious:

$$N = pq \ . \tag{1}$$

Next, since d is the inverse of e modulo $\varphi(N) = (p-1)(q-1) = N - p - q + 1$, we have

$$ed \equiv 1 \pmod{\varphi(N)}$$

and, modulo $p-1$ and $q-1$,

$$ed_p \equiv 1 \pmod{p-1} \qquad \text{and} \qquad ed_q \equiv 1 \pmod{q-1} \ .$$

[2] This is obvious for p and q and well known for d (cf. [7]); d_p reveals p as $\gcd(a^{ed_p-1}-1, N)$ with high probability for random a provided $d_p \neq d_q$, and similarly for d_q; if d_p and d_q are equal to each other then they are also equal to d.

As it happens, it is more convenient for us to write explicitly the terms hidden in the three congruences above, obtaining

$$ed = k(N - p - q + 1) + 1 \tag{2}$$
$$ed_p = k_p(p - 1) + 1 \tag{3}$$
$$ed_q = k_q(q - 1) + 1 \ . \tag{4}$$

It may appear that we have thereby introduced three new unknowns: k, k_p, and k_q. But in fact for small e we can compute each of these three variables given even a badly-degraded version of d.

Computing k. The following argument, due to Boneh, Durfee, and Frankel [4], shows that k must be in the range $0 < k < e$. We know $d < \varphi(N)$. Assume $e \leq k$; then $ed < k\varphi(N) + 1$, which contradicts (2). The case $k = 0$ is also impossible, as can be seen by reducing (2) modulo e. This shows that we can enumerate all possible values of k, having assumed that e is small.

For each such choice k', define

$$\tilde{d}(k') \overset{\text{def}}{=} \left\lfloor \frac{k'(N+1)+1}{e} \right\rfloor \ .$$

As Boneh, Durfee, and Frankel observe, when k' equals k, this gives an excellent approximation for d:

$$0 \leq \tilde{d}(k) - d \leq k(p+q)/e < p + q \ .$$

In particular, when p and q are balanced, we have $p + q < 3\sqrt{N}$, which means that $\tilde{d}(k)$ agrees with d on their $\lfloor n/2 \rfloor - 2$ most significant bits. (Our analysis applies also in the less common case when p and q are unbalanced, but we omit the details.) This means that small-public-exponent RSA leaks half the bits of the private exponent in one of the candidate values $\tilde{d}(1), \ldots, \tilde{d}(e-1)$.

The same fact allows us to go in the other direction, using information about d to determine k, as again noted by Boneh, Durfee, and Frankel. We are given $\tilde{d}$, a corrupted version of d. We enumerate $\tilde{d}(1), \ldots, \tilde{d}(e-1)$ and check which of these agrees, in its more significant half, with the known bits of $\tilde{d}$. Provided that $\delta n/2 \gg \lg e$, there will be just one value of k' for which $\tilde{d}(k')$ matches; that value is k. Even for 1024-bit N and 32-bit e, there is, with overwhelming probability, enough information to compute k for any δ we consider in this paper. This observation has two implications:

1. we learn the correct k used in (2); and
2. we correct the more significant half of the bits of $\tilde{d}$, by copying from $\tilde{d}(k)$.

Computing k_p and k_q. Once we have determined k, we can compute k_p and k_q. First, observe that by an analysis like that above, we can show that $0 < k_p, k_q < e$. This, of course, means that $k_p = (k_p \bmod e)$ and $k_q = (k_q \bmod e)$; when we solve for k_p and k_q modulo e, this will reveal the actual values used in

(3) and (4). Now, reducing equations (1)–(4) modulo e, we obtain the following congruences:

$$N \equiv pq \tag{5}$$
$$0 \equiv k(N - p - q + 1) + 1 \tag{6}$$
$$0 \equiv k_p(p - 1) + 1 \tag{7}$$
$$0 \equiv k_q(q - 1) + 1 \ . \tag{8}$$

These are four congruences in four unknowns: p, q, k_p, and k_q; we solve them as follows. From (7) and (8) we write $(p-1) \equiv -1/k_p$ and $(q-1) \equiv -1/k_q$; we substitute these into the equation obtained from using (5) to reexpress $\varphi(N)$ in (6): $0 \equiv k(N-p-q+1)+1 \equiv k(p-1)(q-1)+1 \equiv k(-1/k_p)(-1/k_q)+1 \equiv k/(k_pk_q)+1$, or

$$k + k_pk_q \equiv 0 \ . \tag{9}$$

Next, we return to (6), substituting in (7), (8), and (9):

$$\begin{aligned} 0 &\equiv k(N - p - q + 1) + 1 \\ &\equiv k(N-1) - k(p - 1 + q - 1) + 1 \\ &\equiv k(N-1) - (-k_pk_q)(-1/k_p - 1/k_q) + 1 \\ &\equiv k(N-1) - (k_q + k_p) + 1 \ ; \end{aligned}$$

we solve for k_p by substituting $k_q = -k/k_p$, obtaining

$$0 \equiv k(N-1) - (k_p - k/k_p) + 1 \ ,$$

or, multiplying both sides by k_p and rearranging,

$$k_p^2 - \big[k(N-1) + 1\big]k_p - k \equiv 0 \ . \tag{10}$$

This congruence is easy to solve modulo e and, in the common case where e is prime, has two solutions, just as it would over $\mathbb{C}$. One of the two solutions is the correct value of k_p; and it is easy to see, by symmetry, that the other must be the correct value of k_q. We need therefore try just two possible assignments to k_p and k_q in reconstructing the RSA key. When e has m distinct prime factors, there may be up to 2^m roots [4].

Note that we also learn the values of p and q modulo e. If we then use the procedure outlined below to decode the r least significant bits of p (up to a list of possibilities), we will know $p \bmod e2^r$; we can then factor N, provided $r + \lg e > n/4$, by applying Boneh, Durfee, and Frankel's Corollary 2.2 ([4]; a generalization of Coppersmith's attack on RSA with known low-order bits [6, Theorem 5] that removes the restriction that the partial knowledge of p must be modulo a power of 2).

3 The Reconstruction Algorithm

Once we have the above relationships between key data, the remainder of the attack consists of enumerating all possible partial keys and pruning those that do not satisfy these constraints. More precisely, given bits 1 through $i-1$ of a potential key, generate all combinations of values for bit i of p, q, d, d_p, d_q, and keep a candidate combination if it satisfies (1), (2), (3), and (4) mod 2^i.

The remainder of this section details how to generate and prune these partial solutions.

In what follows, we assume that we know the values of k_p and k_q. When equation (10) has two distinct solutions, we must run the algorithm twice, once for each of the possible assignments to k_p and k_q.

Let $p\,[i]$ denote the ith bit of p, where the least significant bit is bit 0, and similarly index the bits of q, d, d_p and d_q. Let $\tau(x)$ denote the exponent of the largest power of 2 that divides x.

As p and q are large primes, we know they are odd, so we can correct $p\,[0] = q\,[0] = 1$. It follows that $2 \mid p-1$, so $2^{1+\tau(k_p)} \mid k_p(p-1)$. Thus, reducing (3) modulo $2^{1+\tau(k_p)}$, we have

$$ed_p \equiv 1 \pmod{2^{1+\tau(k_p)}} \ .$$

Since we know e, this allows us immediately to correct the $1+\tau(k_p)$ least significant bits of d_p. Similar arguments using (4) and (2) allow us to correct the $1+\tau(k_q)$ and $2+\tau(k)$ bits of d_q and d, respectively.

What is more, we can easily see that, having fixed bits $< i$ of p, a change in $p\,[i]$ affects d_p not in bit i but in bit $i+\tau(k_p)$; and, similarly, a change in $q\,[i]$ affects $d_q\left[i+\tau(k_q)\right]$, and a change in $p\,[i]$ or $q\,[i]$ affects $d\left[i+\tau(k)\right]$. When any of k, k_p, or k_q is odd, this is just the trivial statement that changing bit i of the right-hand side of an equation changes bit i of the left-hand side. Powers of 2 in k_p shift left the bit affected by $p\,[i]$, and similarly for the other variables.

Having recovered the least-significant bits of each of our five variables, we now attempt to recover the remaining bits. For each bit index i, we consider a slice of bits:

$$p\,[i] \qquad q\,[i] \qquad d\left[i+\tau(k)\right] \qquad d_p\left[i+\tau(k_p)\right] \qquad d_q\left[i+\tau(k_q)\right] \ .$$

For each possible solution up to bit slice $i-1$, generate all possible solutions up to bit slice i that agree with that solution at all but the ith position. If we do this for all possible solutions up to bit slice $i-1$, we will have enumerated all possible solutions up to bit slice i. Above, we already described how to obtain the only possible solution up to $i=0$; this is the solution we use to start the algorithm. The factorization of N will be revealed in one or more of the possible solutions once we have reached $i = \lfloor n/2 \rfloor$.[3]

All that remains is how to lift a possible solution (p', q', d', d'_p, d'_q) for slice $i-1$ to possible solutions for slice i. Naïvely there are $2^5 = 32$ such possibilities, but in fact there are at most 2 and, for large enough δ, almost always fewer.

First, observe that we have four constraints on the five variables: equations (1), (2), (3), and (4). By plugging in the values up to slice $i-1$, we obtain

[3] In fact, as we discussed in Section 2 above, information sufficient to factor N will be revealed much earlier, at $i = \lceil n/4 - \lg e \rceil$.

from each of these a constraint on slice i, namely values $c_1, \ldots, c_4$ such that the following congruences hold modulo 2:

$$\begin{aligned} p\,[i] + q\,[i] &\equiv c_1 \pmod 2 \\ d\,\big[i + \tau(k)\big] + p\,[i] + q\,[i] &\equiv c_2 \pmod 2 \\ d_p\,\big[i + \tau(k_p)\big] + p\,[i] &\equiv c_3 \pmod 2 \\ d_q\,\big[i + \tau(k_q)\big] + q\,[i] &\equiv c_4 \pmod 2 \ . \end{aligned} \tag{11}$$

For example, if N and $p'q'$ agree at bit i, $c_1 = 0$; if not, $c_1 = 1$. Four constraints on five unknowns means that there are exactly two possible choices for bit slice i satisfying these four constraints. (Expressions for the c_is are given in (13).)

Next, it may happen that we know the correct value of one or more of the bits in the slice, through our partial knowledge of the private key. These known bits might agree with neither, one, or both of the possibilities derived from the constraints above. If neither possible extension of a solution up to $i - 1$ agrees with the known bits, that solution is pruned. If δ is sufficiently large, the number of possibilities at each i will be kept small.

4 Algorithm Runtime Analysis

The main result of this section is summarized in the following informal theorem.

Theorem 1. *Given the values of a $\delta = .27$ fraction of the bits of p, q, d, $d \bmod p$, and $d \bmod q$, the algorithm will correctly recover an n-bit RSA key in expected $O(n^2)$ time with probability $1 - \frac{1}{n^2}$.*

The running time of the algorithm is determined by the number of partial keys examined. To bound the total number of keys seen by the program, we will first understand how the structure of the constraints on the RSA key data determines the number of partial solutions generated at each step of the algorithm. Then we will use this understanding to calculate some of the distribution of the number of solutions generated at each step over the randomness of p and q and the missing bits. Finally we characterize the global behavior of the program and provide a bound on the probability that the total number of branches examined over the entire run of the program is too large.

Lifting solutions mod 2^i. The process of generating bit i of a partial solution given bits 0 through $i-1$ can be seen as lifting a solution to the constraint equations mod 2^i to a solution mod 2^{i+1}. Hensel's lemma characterizes the conditions when this is possible.

Lemma 1 (Multivariate Hensel's Lemma). *A root $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ of the polynomial $f(x_1, x_2, \ldots, x_n) \bmod \pi^i$ can be lifted to a root $\mathbf{r} + \mathbf{b} \bmod \pi^{i+1}$ if $\mathbf{b} = (b_1\pi^i, b_2\pi^i, \ldots, b_n\pi^i)$, $0 \le b_j \le \pi - 1$ is a solution to the equation*

$$f(\mathbf{r} + \mathbf{b}) = f(\mathbf{r}) + \sum_j b_j \pi^i f_{x_j}(\mathbf{r}) \equiv 0 \pmod{\pi^{i+1}} \ .$$

(Here, f_{x_j} is the partial derivative of f with respect to x_j.)

We can rewrite the lemma using the notation of Section 3. Write $\mathbf{r}$ in base $\pi = 2$ and assume the i first bits are known. Then the lemma tells us that the next bit of $\mathbf{r}$, $\mathbf{r}[i] = (r_1[i], r_2[i], \ldots)$, must satisfy

$$f(\mathbf{r})[i] + \sum_j f_{x_j}(\mathbf{r}) r_j[i] \equiv 0 \pmod 2 \ . \tag{12}$$

In our case, the constraint polynomials generated in Section 2, equations (1)–(4) form four simultaneous equations in five variables. Given a partial solution (p', q', d', d'_p, d'_q) up to slice i of the bits, we apply the condition in equation (12) above to each polynomial and reduce modulo 2 to obtain the following conditions, modulo 2, on bit i:

$$\begin{aligned}
p\,[i] + q\,[i] &\equiv (n - p'q')\,[i] \\
d\,\big[i+\tau(k)\big] + p\,[i] + q\,[i] &\equiv \big(k(N+1) + 1 - k(p'+q') - ed'\big)\,\big[i+\tau(k)\big] \\
d_p\,\big[i+\tau(k_p)\big] + p\,[i] &\equiv \big(k_p(p'-1) + 1 - ed'_p\big)\,\big[i+\tau(k_p)\big] \\
d_q\,\big[i+\tau(k_q)\big] + q\,[i] &\equiv \big(k_q(q'-1) + 1 - ed'_q\big)\,\big[i+\tau(k_q)\big] \ .
\end{aligned} \tag{13}$$

These are precisely (11).

4.1 Local Branching Behavior

Without additional knowledge of the keys, the system of equations in (13) is underconstrained, and each partial satisfying assignment can be lifted to two partial satisfying assignments for slice i. If bit $i-1$ of a variable x is known, the corresponding $x\,[i-1]$ is fixed to the value of this bit, and the new partial satisfying assignments correspond to solutions of (13) with these bit values fixed. There can be zero, one, or two new solutions at bit i generated from a single solution at bit $i-1$, depending on the known values.

Now that we have a framework for characterizing the partial solutions generated at step i from a partial solution generated at step $i-1$, we will assume that a random fraction δ of the bits of the key values are known, and estimate the expectation and variance of the number of these solutions that will be generated.

In order to understand the number of solutions to the equation, we would like to understand the behavior of the c_i when the partial solution may not be equal to the real solution. Let $\Delta x = x - x'$, then substituting $x' = x - \Delta x$ into (13) we see that any solution to (11) corresponds to a solution to

$$\begin{aligned}
\Delta p\,[i] + \Delta q\,[i] &\equiv (q\Delta p + p\Delta q + \Delta p\Delta q)\,[i] && \pmod 2 \\
\Delta d\,\big[i+\tau(k)\big] + \Delta p\,[i] + \Delta q\,[i] &\equiv (e\Delta d + k\Delta p + k\Delta q))\,\big[i+\tau(k)\big] && \pmod 2 \\
\Delta d_p\,\big[i+\tau(k_p)\big] + \Delta p\,[i] &\equiv (e\Delta d_p - k_p\Delta p)\,\big[i+\tau(k_p)\big] && \pmod 2 \\
\Delta d_q\,\big[i+\tau(k_q)\big] + \Delta q\,[i] &\equiv (e\Delta d_q - k_q\Delta q)\,\big[i+\tau(k_q)\big] && \pmod 2
\end{aligned}$$

and $\Delta x\,[i]$ is restricted to 0 if bit i of x is fixed.

Incorrect solutions generated from a correct solution. When the partial satisfying assignment is correct, all of the Δx will be equal to 0. If all of the $\Delta x\,[i]$ are unconstrained or if only $\Delta d\,[i+\tau(k)]$ is set to 0, there will be two possible solutions (of which we know one is "good" and the other is "bad"), otherwise there will be a single good solution. Let Z_g be a random variable denoting the number of bad solutions at bit $i+1$ generated from a single good solution at bit i. Since each $\Delta x\,[i]$ is set to 0 independently with probability δ, the expected number of bad solutions generated from a good solution is equal to

$$\mathrm{E}\,Z_g = \delta(1-\delta)^4 + (1-\delta)^5 \qquad \text{and} \qquad \mathrm{E}\,Z_g^2 = \mathrm{E}\,Z_g \ .$$

Both these expressions are dependent only on δ.

Incorrect solutions generated from an incorrect solution. When the partial satisfying assignment is incorrect, at least one of the Δx is nonzero. The expected number of new incorrect satisfying assignments generated from an incorrect satisfying assignment is dependent both on δ and on the behavior of the b_j.

We conjecture the following is close to being true:

Conjecture 1. For random p and q and for Δx not all zero and satisfying

$$\begin{aligned} q\Delta p + p\Delta q - \Delta p\Delta q &= 0 \pmod{2^i} \\ e\Delta d + k\Delta p + k\Delta q &= 0 \pmod{2^{i+\tau(k)}} \\ e\Delta d_p - k_p\Delta p &= 0 \pmod{2^{i+\tau(k_p)}} \\ e\Delta d_q - k_q\Delta q &= 0 \pmod{2^{i+\tau(k_q)}} \ , \end{aligned}$$

the next bit of each congruence is 0 or 1 independently with probability near 1/2.

We tested this empirically; each value of the vector (b_1, b_2, b_3, b_4) occurs with probability approximately 1/16. (The error is approximately 5% for $\delta = 0.25$ and $n = 1024$, and approximately 2% for $\delta = 0.25$ and $n = 4096$.)

Let W_b be a random variable denoting the number of bad solutions at bit $i+1$ generated from a single bad solution at bit i. Assuming Conjecture 1,

$$\mathrm{E}\,W_b = \frac{(2-\delta)^5}{16} \qquad \text{and} \qquad \mathrm{E}\,W_b^2 = \mathrm{E}\,W_b + \delta(1-\delta)^4 + 2(1-\delta)^5 \ .$$

Note that the expectation is over the randomness of p and q and the positions of the unknown bits of the key.

When partial knowledge of some of the values (p, q, d, d_p, d_q) is totally unavailable, we can obtain a similar expression.

4.2 Global Branching Behavior at Each Step of the Program

Now that we have characterized the effect that the constraints have on the branching behavior of the program, we can abstract away all details of RSA entirely and examine the general branching process of the algorithm. We are able to characterize the behavior of the algorithm, and show that if the expected number of branches from any partial solution to the program is less than one, then the total number of branches examined at any step of the program is expected to be constant. All of the following analysis assumes Conjecture 1.

Let X_i be a random variable denoting the number of bad assignments at step i, and recall that Z_g and W_b are random variables denoting the number of bad solutions at bit $i+1$ generated from a single good or bad solution at bit i.

Theorem 2

$$\mathrm{E}\, X_i = \frac{\mathrm{E}\, Z_g}{1 - \mathrm{E}\, W_b}(1 - (\mathrm{E}\, W_b)^i)$$

This expression can be calculated in a number of ways; we demonstrate how to do so using generating functions in Appendix A.

When $\mathrm{E}\, W_b < 1$, we can bound $\mathrm{E}\, X_i$ from above.

$$\mathrm{E}\, X_i \le \frac{\mathrm{E}\, Z_g}{1 - \mathrm{E}\, W_b}$$

In the previous section, we calculated expressions for $\mathrm{E}\, Z_g$ and $\mathrm{E}\, W_b$ dependent only on δ, thus when $\mathrm{E}\, W_b < 1$, $\mathrm{E}\, X_i$ can be bounded above by a constant dependent on δ and not on i.

We can evaluate this expression numerically using the values for the expected number of bad solutions discovered in the last section.

In the case with four equations and five unknowns (that is, we have partial knowledge of p, q, d, d_p, and d_q), $\mathrm{E}\, W_b < 1$ at $\delta > 2 - 2^{\frac{4}{5}}$. For $\delta = .2589$, $\mathrm{E}\, X_i < 93247$; for $\delta = .26$, $\mathrm{E}\, X_i < 95$; and for $\delta = .27$ $\mathrm{E}\, X_i < 9$.

In a similar fashion we can obtain the following complicated expression for the variance $\mathrm{Var}\, X_i = \mathrm{E}\, X^2 - (\mathrm{E}\, X)^2$.

Theorem 3

$$\mathrm{Var}\, X_i = \alpha_1 + \alpha_2(\mathrm{E}\, W_b)^i + \alpha_3(\mathrm{E}\, W_b)^{2i} \tag{14}$$

with

$$\begin{aligned}
\alpha_1 &= \frac{\mathrm{E}\, Z_g \,\mathrm{Var}\, W_b + (1 - \mathrm{E}\, W_b)\,\mathrm{Var}\, Z_g}{(1 - (\mathrm{E}\, W_b)^2)(1 - \mathrm{E}\, W_b)} \\
\alpha_2 &= \frac{\mathrm{E}\, W_b^2 + \mathrm{E}\, W_b - 2\,\mathrm{E}\, W_b\,\mathrm{E}\, Z_g - \mathrm{E}\, Z_g}{1 - \mathrm{E}\, W_b} + 2\left(\frac{\mathrm{E}\, Z_g}{1 - \mathrm{E}\, W_b}\right)^2 \\
\alpha_3 &= -\alpha_1 - \alpha_2 \ .
\end{aligned}$$

Again evaluating numerically for five unknowns and four equations, at $\delta = .26$ $\mathrm{Var}\, X_i < 7937$, at $\delta = .27$ $\mathrm{Var}\, X_i < 80$, and at $\delta = .28$ $\mathrm{Var}\, X_i < 23$.

4.3 Bounding the Total Number of Keys Examined

Now that we have some information about the distribution of the number of partial keys examined at each step, we would like to understand the distribution of the total number of keys examined over an entire run of the program.

We know the expected total number of keys examined for an n-bit key is

$$\mathrm{E}\left[\sum_{i=0}^{n} X_i\right] \leq \frac{\mathrm{E}\, Z_g}{1-\mathrm{E}\, W_b} n \ .$$

We will bound how far the total sum is likely to be from this expectation. First, we apply the following bound on the variance of a sum of random variables:

Lemma 2

$$\operatorname{Var}\sum_{i=1}^{n} X_i \leq n^2 \max_i \operatorname{Var} X_i$$

The proof writes the variance of the sum in terms of covariance, and applies Schwartz's inequality and $\sqrt{ab} \leq \frac{a+b}{2}$.

Apply Chebyshev's inequality to bound the likelihood that $\sum X_i$ is too large:

$$\Pr(|\textstyle\sum_i X_i - \mathrm{E}\sum_i X_i| \geq n\alpha) \leq \frac{1}{(n\alpha)^2} \operatorname{Var} \textstyle\sum_i X_i \ .$$

Apply the above lemma to obtain

$$\Pr(|\textstyle\sum_i X_i - \mathrm{E}\sum_i X_i| \geq n\alpha) \leq \frac{1}{\alpha^2} \max_i \operatorname{Var} X_i \ .$$

When $\delta = .27$, setting $\alpha > 9n$ gives that, for an n-bit key, the algorithm will examine more than $9n^2 + 71n$ potential keys with probability less than $\frac{1}{n^2}$.

4.4 Missing Key Fields

The same results apply when we have partial knowledge of fewer key fields.

– If the algorithm has partial knowledge of d, p, and q but no information on d_p and d_q, we know that

$$\mathrm{E}\, Z_g = \delta(1-\delta)^2 + (1-\delta)^3 \qquad \mathrm{E}\, Z_g^2 = \mathrm{E}\, Z_g$$
$$\mathrm{E}\, W_b = \frac{(2-\delta)^3}{4} \qquad \mathrm{E}\, W_b^2 = \mathrm{E}\, W_b + \delta(1-\delta)^2 + 2(1-\delta)^3 \ ,$$

so $\mathrm{E}\, W_b < 1$ when $\delta > 2 - 2^{\frac{3}{4}} \approx .4126$. Then for $\delta = .42$ the probability that the algorithm examines more than $22n^2 + 24n$ keys is less than $\frac{1}{n^2}$.

– If the algorithm has partial knowledge of p and q but no information on the other values,

$$\mathrm{E}\, Z_g = (1-\delta)^2 \qquad \mathrm{E}\, Z_g^2 = \mathrm{E}\, Z_g$$
$$\mathrm{E}\, W_b = \frac{(2-\delta)^2}{2} \qquad \mathrm{E}\, W_b^2 = \mathrm{E}\, W_b + 2(1-\delta)^2 \ .$$

Then $\mathrm{E}\, W_b < 1$ when $\delta > 2 - 2^{\frac{1}{2}} \approx .5859$. When $\delta = .59$ the probability that the algorithm examines more than $29n^2 + 29n$ keys is less than $\frac{1}{n^2}$.

5 Implementation and Performance

We have developed an implementation of our algorithm in approximately 850 lines of C++, using NTL version 5.4.2 and GMP version 4.2.2. Our tests were run, in 64-bit mode, on an Intel Core 2 Duo processor at 2.4 GHz with 4 MB of L2 cache and 4 GB of DDR2 SDRAM at 667 MHz on an 800 MHz bus.

We ran experiments for key sizes between 512 bits and 8192 bits, and for δ values between 0.40 and 0.24. The public exponent is always set to 65537. In each experiment, a key of the appropriate size is randomly censored so that exactly a δ fraction of the bits of the private key components considered together is available to be used for reconstruction. To reduce the time spent on key generation, we reused keys: We generated 100 keys for each key size. For every δ and keysize, we ran 100 experiments with each one of the pregenerated keys, for a total of 10,000 experimental runs. In all, we conducted over 1.1 million runs.

For each run, we recorded the *length* and *width*. The length is the total number of keys considered in the run of the algorithm, at all bit indices; the width is the maximum number of keys considered at any single bit index. These correspond essentially to $\sum_{i=1}^{n/2} X_i$ and $\max_i X_i$, in the notation of Section 4, but can be somewhat larger because we run the algorithm twice in parallel to account for both possible matchings of solutions of (10) to k_p and k_q. To avoid thrashing, we killed runs as soon as the width for some index i exceeded 1,000,000.

When the panic width was not exceeded, the algorithm always ran to completion and correctly recovered the factorization of the modulus.

Of the 900,000 runs of our algorithm with $\delta \geq 0.27$, only a single run ($n = 8192$, $\delta = 0.27$) exceeded the panic width. Applying a Chebyshev bound in this case (with $\mathrm{E}\, X_i = 9$ and $\mathrm{Var}\, X_i = 80$) suggests that a width of 1,000,000 should happen with extremely low probability.

Even below $\delta = 0.27$, our algorithm almost always finished within the allotted time. Table 1 shows the number of runs (out of 10,000) in which the panic width was exceeded for various parameter settings. Even for $n = 8192$ and $\delta = 0.24$, our algorithm recovered the factorization of the modulus in more than 97% of all runs. And in many of the overly long runs, the number of bits recovered before the panic width was exceeded suffices to allow recovering the rest using the lattice methods considered in Section 2; this is true of 144 of the 274 very long runs at $n = 8192$ and $\delta = 0.24$, for example.

Table 1. Runs (out of 10,000) in which width exceeded 1,000,000

	$n = 512$	768	1024	1536	2048	3072	4096	6144	8192
$\delta = 0.27$	0	0	0	0	0	0	0	0	1
0.26	0	0	0	0	1	5	3	4	8
0.25	0	0	3	6	8	10	17	35	37
0.24	4	5	7	27	50	93	121	201	274

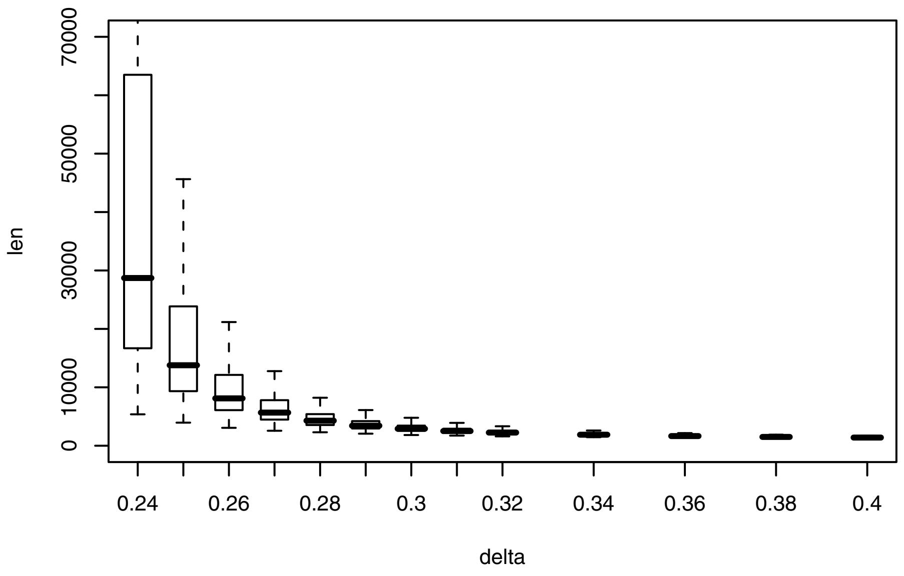


Fig. 1. Boxplot for total number of keys examined by algorithm for $n = 2048$, varying δ

As expected, search runtime was essentially linear in the total number of keys examined. For $n = 1024$, for example, examining a single key took approximately 5 μsec; for $n = 6144$, approximately 8 μsec. The setup time varied depending on whether k was closer to 0 or to e, but never exceeded 210 msec, even for $n = 8192$.

The plot in Figure 1 gives the behavior for $n = 2048$. For each value of δ we show, using a boxplot, the distribution of the total number of keys examined by runs of the algorithm – i.e., the length of the run. (In our boxplot, generated using R's `boxplot` function, the central bar corresponds to the median, the hinges to the first and third quartiles, and the whisker extents depend on the interquartile range.)

In the full version of this paper [11] we undertake additional analysis of the runtime data.

Acknowledgments

We thank Dan Boneh for suggesting the connection to Hensel lifting; Amir Dembo for improving our branching process analysis; Daniele Micciancio for extensive discussions on using lattice reduction to solve the knapsack problem implicit in our attack; and Eric Rescorla for his help with analyzing the observed runtimes of our algorithm.

In addition, we had fruitful discussions with J. Alex Halderman, Howard Karloff, and N. J. A. Sloane. We would also like to thank the anonymous Crypto reviewers for their comments and suggestions.

This material is based in part upon work supported by the National Science Foundation under CNS grant no. 0831532 (Cyber Trust) and a Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Wichs, D.: Public key cryptography in the bounded retrieval model and security against side-channel attacks. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–53. Springer, Heidelberg (2009)
3. Boneh, D.: Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society (AMS) 46(2), 203–213 (1999)
4. Boneh, D., Durfee, G., Frankel, Y.: An attack on RSA given a small fraction of the private key bits. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 25–34. Springer, Heidelberg (1998)
5. Boneh, D., Shacham, H.: Fast variants of RSA. RSA Cryptobytes 5(1), 1–9 (Winter/Spring 2002)
6. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptology 10(4), 233–260 (1997)
7. Coron, J.-S., May, A.: Deterministic polynomial-time equivalence of computing the RSA secret key and factoring. J. Cryptology 20(1), 39–50 (2007)
8. Dodis, Y., Tauman Kalai, Y., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) Proceedings of STOC 2009. ACM Press, New York (2009)
9. Goldwasser, S.: Cryptography without (hardly any) secrets. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 369–370. Springer, Heidelberg (2009)
10. Halderman, J.A., Schoen, S., Heninger, N., Clarkson, W., Paul, W., Calandrino, J., Feldman, A., Appelbaum, J., Felten, E.: Lest we remember: Cold boot attacks on encryption keys. In: Van Oorschot, P. (ed.) Proceedings of USENIX Security 2008, July 2008, pp. 45–60. USENIX (2008)
11. Heninger, N., Shacham, H.: Reconstructing RSA private keys from random key bits. Cryptology ePrint Archive, Report 2008/510 (December 2008), `http://eprint.iacr.org/`
12. Herrmann, M., May, A.: Solving linear equations modulo divisors: On factoring given any bits. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 406–424. Springer, Heidelberg (2008)
13. Karlin, S., Taylor, H.M.: A First Course in Stochastic Processes. Academic Press, London (1975)
14. Maurer, U.: On the oracle complexity of factoring integers. Computational Complexity 5(3/4), 237–247 (1995)
15. May, A.: New RSA Vulnerabilities Using Lattice Reduction Methods. PhD thesis, University of Paderborn (October 2003)
16. May, A.: Using LLL-reduction for solving RSA and factorization problems: A survey. In: Nguyen, P. (ed.) Proceedings of LLL+25 (June 2007)
17. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)

18. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
19. Nguyen, P., Stern, J.: Adapting density attacks to low-weight knapsacks. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 41–58. Springer, Heidelberg (2005)
20. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
21. Rivest, R., Shamir, A.: Efficient factoring based on partial information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986)
22. RSA Laboratories. PKCS #1 v2.1: RSA cryptography standard (June 2002), `http://www.rsa.com/rsalabs/node.asp?id=2125`
23. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S.: When private keys are public: Results from the 2008 Debian OpenSSL debacle (May 2009) (manuscript)

A Computing the Expectation and Variance

In this appendix, we derive expressions for the expectation and variance of the number of incorrect keys generated at each step of the program. Let X_i be a random variable denoting the number of bad assignments at step i. We will calculate the expectation $\mathrm{E}\,X_i$ and variance $\mathrm{Var}\,X_i$. (We know that the number of good assignments is always equal to one.)

To calculate these values, we will use probability generating functions. For more information on this approach, see e.g., [13, Ch. 8]. A probability generating function $F(s) = \sum \Pr[X = k]s^k$ represents the distribution of the discrete random variable X. $F(s)$ satisfies the following identities:

$$F(1) = 1 \ , \qquad \mathrm{E}\,X = F'(1) \ , \qquad \text{and} \qquad \mathrm{Var}\,X = F''(1) + F'(1) - F'(1)^2 \ .$$

Let $G_i(s)$ be the probability generating function for the X_i, $z(s)$ the probability generating function for the Z_g (the number of bad assignments generated from a correct assignment) and $w(s)$ the probability generating function for the W_b (the number of bad assignments generated from a bad assignment).

From Section 4, we know that

$$z'(1) = \mathrm{E}\,Z_g \ , \qquad z''(1) = \mathrm{E}\,Z_g^2 - \mathrm{E}\,Z_g \ ,$$
$$w'(1) = \mathrm{E}\,W_b \ , \qquad \text{and} \qquad w''(1) = \mathrm{E}\,W_b^2 - \mathrm{E}\,W_b \ .$$

Expectation of X_i. We will calculate $\mathrm{E}\,X_i = G_i'(1)$. $G_i(s)$ satisfies the recurrence

$$G_{i+1}(s) = G_i(w(s))z(s) \ , \tag{15}$$

that is, that the number of bad solutions at each step is equal to the number of bad solutions lifted from bad solutions plus the number of bad solutions produced from good solutions. (Recall that a generating function for the sum of two independent random variables is given by the convolution of their generating functions.) We also have that

$$G_0(s) = 1 \ ,$$

because initially there are no bad solutions. Differentiating (15) gives

$$G_i'(s) = (G_{i-1}(w(s))w'(s)z(s) + G_{i-1}(w(s))z'(s) \ . \tag{16}$$

Set $s = 1$ and use the fact that $G_i(1) = w(1) = z(1) = 1$ to obtain

$$G_i'(1) = w'(1)G_{i-1}'(1) + z'(1) \ .$$

Solving the recurrence yields

$$G_i'(1) = \frac{z'(1)}{1 - w'(1)}(1 - (w'(1))^i) \ . \tag{17}$$

If $w'(1) < 1$, then $w'(1)^i$ tends to 0 as i increases and

$$EX_i = G_i'(1) < \frac{z'(1)}{1 - w'(1)} \tag{18}$$

for all i. The expected number of bad solutions at any step of the process will be bounded by a value dependent only on δ and not on i.

Variance of X_i. To compute the variance $\operatorname{Var} X_i = G_i''(1) + G_i'(1) - (G_i'(1))^2$, we differentiate (16) again to obtain

$$\begin{aligned} G_i''(s) = G_{i-1}''(w(s))w'(s)w'(s)z(s) + G_{i-1}'(w(s))w''(s)z(s) \\ + 2G_{i-1}'(w(s))w'(s)z'(s) + G_{i-1}(w(s))z''(s) \ . \end{aligned} \tag{19}$$

Evaluating at $s = 1$ gives

$$G_i''(1) = G_{i-1}''(1)w'(1)^2 + G_{i-1}'(1)w''(1) + 2G_{i-1}'(1)w'(1)z'(1) + z''(1) \ .$$

Substitute in (17) to get

$$\begin{aligned} G_i''(1) = G_{i-1}''(1)w'(1)^2 + \frac{z'(1)}{1 - w'(1)}(1 - (w'(1))^i)w''(1) \\ + 2\frac{z'(1)}{1 - w'(1)}(1 - (w'(1))^i)w'(1)z'(1) + z''(1) \ . \end{aligned} \tag{20}$$

The general solution to this recurrence is

$$G_i''(1) = c_1 + c_2 w'(1)^i + c_3 w'(1)^{2i} \tag{21}$$

with

$$\begin{aligned} c_1 &= \frac{1}{1 - w'(1)^2}\left(\frac{z'(1)}{1 - w'(1)}(w''(1) + 2w'(1)z'(1)) + z''(1)\right) \\ c_2 &= -\frac{1}{1 - w'(1)}(w''(1) + 2w'(1)z'(1)) \\ c_3 &= -c_1 - c_2 \ . \end{aligned}$$

Public-Key Cryptosystems Resilient to Key Leakage[⋆]

Moni Naor[⋆⋆] and Gil Segev[⋆⋆⋆]

Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel
{moni.naor,gil.segev}@weizmann.ac.il

Abstract. Most of the work in the analysis of cryptographic schemes is concentrated in abstract adversarial models that do not capture *side-channel attacks*. Such attacks exploit various forms of unintended information leakage, which is inherent to almost all physical implementations. Inspired by recent side-channel attacks, especially the "cold boot attacks", Akavia, Goldwasser and Vaikuntanathan (TCC '09) formalized a realistic framework for modeling the security of encryption schemes against a wide class of side-channel attacks in which adversarially chosen functions of the secret key are leaked. In the setting of public-key encryption, Akavia et al. showed that Regev's lattice-based scheme (STOC '05) is resilient to any leakage of $L/\text{polylog}(L)$ bits, where L is the length of the secret key.

In this paper we revisit the above-mentioned framework and our main results are as follows:

- We present a generic construction of a public-key encryption scheme that is resilient to key leakage from any *universal hash proof system*. The construction does not rely on additional computational assumptions, and the resulting scheme is as efficient as the underlying proof system. Existing constructions of such proof systems imply that our construction can be based on a variety of number-theoretic assumptions, including the decisional Diffie-Hellman assumption (and its progressively weaker d-Linear variants), the quadratic residuosity assumption, and Paillier's composite residuosity assumption.
- We construct a new hash proof system based on the decisional Diffie-Hellman assumption (and its d-Linear variants), and show that the resulting scheme is resilient to any leakage of $L(1-o(1))$ bits. In addition, we prove that the recent scheme of Boneh et al. (CRYPTO '08), constructed to be a "circular-secure" encryption scheme, is resilient to any leakage of $L(1-o(1))$ bits. These two proposed schemes complement each other in terms of efficiency.

[⋆] Due to space limitations we refer the reader to a longer version available as [28].

[⋆⋆] Incumbent of the Judith Kleeman Professorial Chair. Research supported in part by a grant from the Israel Science Foundation.

[⋆⋆⋆] Research supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities, and by a grant from the Israel Science Foundation.

– We extend the framework of key leakage to the setting of chosen-ciphertext attacks. On the theoretical side, we prove that the Naor-Yung paradigm is applicable in this setting as well, and obtain as a corollary encryption schemes that are CCA2-secure with any leakage of $L(1 - o(1))$ bits. On the practical side, we prove that variants of the Cramer-Shoup cryptosystem (along the lines of our generic construction) are CCA1-secure with any leakage of $L/4$ bits, and CCA2-secure with any leakage of $L/6$ bits.

1 Introduction

Proving the security of a cryptographic scheme consists of two main ingredients: (1) an *adversarial model* that specifies the adversarial access to the system and the adversary's computational capabilities, and (2) a *notion of security* that specifies what it means to "break" the security of the scheme. Whereas notions of security have significantly evolved over the years (following the seminal work of Goldwasser and Micali [15]), the vast majority of cryptographic schemes are analyzed in abstract adversarial models that do not capture *side-channel attacks*. Such attacks exploit unintended leakage of information which is inherent to almost all physical implementations. Over the years side-channel attacks exposed crucial vulnerabilities of systems that are considered secure in standard adversarial models (see, for example, [3,4,23,24]).

Countermeasures for protecting against side-channel attacks are taken on two complementing levels: the hardware level and the software level. Preventing unintended leakage on the hardware level is typically rather inefficient and expensive, and is even impossible in some cases. It is thus highly desirable to protect, as much as possible, against side-channel attacks on the software level by modeling such attacks using abstract notions of computation.

Physically observable cryptography. In their pioneering work, Micali and Reyzin [27] put forward a powerful and comprehensive framework for modeling security against side-channel attacks. Their framework captures any such attack in which leakage of information occurs as a result of *computation*. The framework relies on the basic assumption that *computation and only computation leaks information*, that is, there is no leakage of information in the absence of computation. This assumption has led to the construction of various cryptographic primitives that are robust to "computational" leakage (see, for example, [14,16,27,30,31]).

Memory-leakage attacks. Recently, Halderman et al. [18] presented a suite of attacks that violate the basic assumption underlying the framework of Micali and Reyzin. Halderman et al. showed that, contrary to popular assumptions, a computer's memory is not erased when it loses power. They demonstrated that ordinary DRAMs typically lose their contents gradually over a period of seconds, and that residual data can be recovered using simple, non-destructive techniques that require only momentary physical access to the machine. Halderman et al. presented attacks that exploit DRAM remanence effects to recover cryptographic keys held in memory. Specifically, their "cold boot" attacks showed that a significant fraction of the bits of a cryptographic key can be recovered if the key is ever stored in memory. Halderman et al. managed to completely compromise the security of several popular disk encryption systems (including BitLocker, TrueCrypt, and FileVault), and to reconstruct DES, AES, and RSA keys (see also the improved RSA key reconstruction by Heninger and Shacham [19]).

Inspired by the cold boot attacks, Akavia, Goldwasser and Vaikuntanathan [2] formalized a general framework for modeling "memory attacks" in which adversarially chosen functions of the secret key are leaked in an adaptive fashion, with the only restriction that the total amount of leakage is bounded. Akavia et al. showed that the lattice-based public-key encryption scheme of Regev [32] is resilient to such key leakage (to an extent that depends on the amount of leakage) by slightly strengthening the computational assumption that is required by the original scheme.

1.1 Our Contributions

In this work we revisit the framework of key-leakage attacks introduced by Akavia et al. in the setting of public-key encryption. We present a generic construction of a public-key encryption scheme that is resilient to key leakage, and show that the construction can be based on a variety of number-theoretic assumptions (see below). Moreover, we demonstrate that our approach leads to encryption schemes that are both resilient to significantly large amounts of leakage, and that are very efficient and can be used in practice (see, in particular, the instantiation in Section 4 that is based on the decisional Diffie-Hellman assumption). In addition, we extend the framework of key-leakage attacks to the setting of chosen-ciphertext attacks. We present both a generic transformation from chosen-plaintext security to chosen-ciphertext security in the context of key-leakage attacks, and efficient schemes that are based on specific number-theoretic assumptions.

In what follows we present a more elaborated exposition of our results, but first, we briefly describe the framework of Akavia et al. and their results. Informally, an encryption scheme is resilient to key-leakage attacks if it is semantically secure even when the adversary obtains sensitive leakage information. This is modeled by providing the adversary with access to a leakage oracle: the adversary can submit any function f and receive $f(sk)$, where sk is the secret key (we note that the leakage functions can be chosen depending on the public key, which is known to the adversary). The adversary can query the leakage oracle adaptively, with only one restriction: the sum of output lengths of all the leakage functions has to be bounded by a predetermined parameter λ (clearly, λ has to be less than the length of the secret key)[1]. A formal definition is provided in Section 3. Akavia et al. showed that Regev's public-key encryption scheme is resilient to any key leakage of $L/\text{polylog}(L)$ bits, where L is the length of the

[1] Akavia et al. refer to such attacks as *adaptive memory attacks*. They also define the notion of *non-adaptive memory attacks* which we discuss later on.

secret key (see improvements to the allowed amount of leakage in the full version of their paper). We are now ready to state our results more clearly:

A generic construction. We present a generic construction of a public-key encryption scheme that is resilient to key leakage from any *universal hash proof system*, a very useful primitive introduced by Cramer and Shoup [7]. The construction does not rely on additional computational assumptions, and the resulting scheme is as efficient as the underlying proof system. Existing constructions of such proof systems [7,22,34] imply that our construction can be based on a variety of number-theoretic assumptions, including the decisional Diffie-Hellman (DDH) assumption and its progressively weaker d-Linear variants, the quadratic residuosity assumption, and Paillier's composite residuosity assumption. The natural approach for achieving security against partial key leakage is to add redundancy to the private key, so that every (short) function of it will still keep many possibilities for the "real secret". Hash proof systems yield a convenient method for doing just that.

We then emphasize a specific instantiation with a simple and efficient DDH-based hash proof system. The resulting encryption scheme is resilient to any leakage of $L(1/2 - o(1))$ bits, where L is the length of the secret key. Although one can instantiate our construction with any hash proof system, we find this specific instantiation rather elegant.

The schemes that result from our generic construction satisfy in fact a more general notion of leakage resilience: these schemes are secure even if the leakage functions chosen by the adversary are applied to the random bits used by the key generation algorithm. This clearly generalizes the framework of Akavia et al. and guarantees security even in case that intermediate values from the process of generating the secret and public keys are leaked[2]. In addition, we consider several other generalizations of the framework of Akavia et al. that are satisfied by our schemes. These include a scenario in which the adversary obtains a noisy version of all of the memory (as in the attack of Halderman et al. [18]), a scenario in which partial results of the decryption process are leaked, and more.

Improved key-leakage resilience. We propose two public-key encryption schemes that are resilient to any key leakage of $L(1 - o(1))$ bits, where L is the length of the secret key. Our proposals are based on the observation that our generic construction from hash proof systems can in fact be based on hash proof systems with a slightly weaker universality property. When viewing hash proof systems as key-encapsulation mechanisms, relaxing the universality property enables us to achieve essentially the best possible ratio between the length of the secret key and the length of the encapsulated symmetric key. This ratio

[2] We note that it is not clear that Regev's scheme is resilient to leakage of intermediate key-related values, or at least, the proof of security of Akavia et al. does not seem to generalize to this setting. The main reason is that their proof of security involves an indistinguishability argument over the public key, and an adversary that has access to the randomness of the key generation algorithm (via leakage queries) can identify that the public key was not sampled from its specified distribution.

translates to the relative amount of key leakage to which the encryption schemes are resilient[3].

For our first proposal we construct a new hash proof system based on the decisional Diffie-Hellman assumption (and more generally, on any of the d-Linear assumptions) that satisfies this weaker universality property. The resulting encryption scheme is then obtained by instantiating our generic construction with this hash proof system. For our second proposal, we show the recent "circular-secure" encryption scheme of Boneh et al. [5] fits into our generic approach using a different hash proof system (that satisfies the same weaker universality property). We then compare our two proposals both conceptually and practically, indicating that they complement each other in terms of efficiency.

Chosen-ciphertext security. We extend the framework of key leakage to the setting of chosen-ciphertext security. Technically, this is a very natural extension by providing the adversary with access to both a leakage oracle and a decryption oracle. On the theoretical side, we show that the Naor-Yung "double encryption" paradigm [12,29] can be used as a general transformation from chosen-plaintext security to chosen-ciphertext security in the presence of key leakage. As an immediate corollary of our above-mentioned results, we obtain a scheme that is CCA2-secure with any leakage of $L(1 - o(1))$ bits, where L is the length of the secret key.

The schemes resulting from the Naor-Yung paradigm are rather inefficient due to the usage of generic non-interactive zero-knowledge proofs. To complement this situation, on the practical side, we prove that variants of the Cramer-Shoup cryptosystem [8] (along the lines of our generic transformation from hash proof systems) are CCA1-secure with any leakage of $L(1/4 - o(1))$ bits, and CCA2-secure with any leakage of $L(1/6 - o(1))$ bits. It is left as an open problem to construct a practical CCA-secure scheme that is resilient to any leakage of $L(1 - o(1))$ bits (where a possible approach is to examine recent refinements of the Cramer-Shoup cryptosystem [1,22,25]).

"Weak" key-leakage security. Akavia et al. also considered the following weaker notion of key leakage (which they refer to as "non-adaptive" leakage): a leakage function f with output length λ is chosen by the adversary ahead of time (without any knowledge of the public key), and then the adversary is given $(pk, f(sk))$. That is, in a "weak" key-leakage attack the leakage function f is chosen independently of pk. Akavia et al. proved that Regev's encryption scheme is resilient to any weak key leakage of $L(1 - o(1))$ bits.

Although this notion of key leakage seems rather limited, it still captures many realistic attacks in which the leakage does not depend on the parameters of the encryption scheme. Specifically, this notion captures the cold boot attack of Halderman et al. [18], in which the leakage depends only on the properties of the hardware devices that are used for storing the secret key.

For weak key-leakage attacks we present a generic construction that transforms any encryption scheme to one that is resilient to any weak leakage of

[3] We do not argue that such a relaxation is in fact necessary for achieving the optimal ratio.

$L(1 - o(1))$ bits, where L is the length of the secret key. The resulting scheme is essentially as efficient as the original one, and does not rely on additional computational assumptions. Our approach crucially relies on the fact that the leakage is independent of the public key. One may interpret our construction as evidence to the deficiency of this weaker notion of key-leakage attacks.

1.2 Related Work

Extensive work has been devoted for protecting against side-channel attacks, and for exploiting side-channels to compromise the security of cryptographic schemes. It is far beyond the scope of this paper to present an exhaustive overview of this ever-growing line of work. We focus here on the results that are most relevant to our work. Already in 1985 Rivest and Shamir [33] introduced a model for leakage attacks in the context of factoring. They considered a scenario in which an adversary is interested in factoring an n-bit modulus $N = PQ$, and is allowed to ask a certain number of arbitrary "Yes/No" questions. Rivest and Shamir asked the following question: how many questions are needed in order to factor N in polynomial time? Clearly, if the adversary is allowed to ask about $n/2$ questions, then the binary representation of P can be fully revealed. Rivest and Shamir showed an attack that requires only $n/3$ questions. Specifically, in their attack the adversary requests the top $n/3$ bits of P. This was later improved by Maurer [26] who showed that ϵn questions are sufficient, for any constant $\epsilon > 0$.

Canetti et al. [6] introduced the notion of *exposure resilient* cryptographic primitives, which remain secure even if an adversary is able to learn almost all of the secret key of the primitive. Most notably, they introduced the notion of an *exposure resilient function*: a deterministic function whose output appears random even if almost all the bits of the input are known (see also the work of Dodis et al. [10] on adaptive security of such functions). Ishai et al. [20,21] considered the more general problem of protecting privacy in circuits, where the adversary can access a bounded number of wires in the circuit. Ishai et al. proposed several techniques for dealing with this type of attacks.

Dziembowski and Pietrzak [14] and Pietrzak [31] introduced a general framework for leakage-resilient cryptography, following the assumption of Micali and Reyzin that only computation leaks information. Their main contributions are constructions of leakage-resilient stream-ciphers. Informally, their model considers cryptographic primitives that proceed in rounds, and update their internal state after each round. In each round, the adversary can obtain bounded leakage information from the portions of memory that were accessed during that round.

Dodis, Tauman Kalai, and Lovett [11] studied the security of symmetric-key encryption schemes under key leakage attacks. They considered leakage of the form $f(sk)$, where sk is the secret key and f is any exponentially-hard one-way function. On one hand they do not impose any restriction on the min-entropy of the secret key given the leakage, but on the other hand, they require that the leakage is a function that is extremely hard to invert. Dodis et al. introduced a new computational assumption that is a generalization of learning parity with noise, and constructed symmetric-key encryption schemes that are resilient to any key leakage that is exponentially hard to invert.

In a concurrent and independent work, Tauman Kalai and Vaikuntanathan [35] considered leakage of hard-to-invert functions in the setting of public-key encryption. Their main result is that the circular-secure encryption scheme of Boneh et al. [5] is resilient to key leakage not only when the secret key has sufficient min-entropy given the leakage function (as also shown in this paper in Section 5.2 as a specific instantiation of our generic approach), but also when the leakage function is exponentially hard to invert. In addition, they proved that the Naor-Yung paradigm can be used to achieve chosen-ciphertext security in the setting of key leakage, and their construction is essentially identical to our construction (see [28, Section 6.1]).

1.3 Paper Organization

The remainder of the paper is organized as follows. In Section 2 we present several notions and tools that are used in our constructions. In Section 3 we formally describe the framework of key-leakage attacks. In Section 4 we present our generic construction from hash proof systems, and provide a simple and efficient instantiation. In Section 5 we present our two proposals that are resilient to any key leakage of $L(1-o(1))$ bits, and provide a comparison between them. In Section 6 we present several generalizations of the framework considered in this paper that are satisfied by our schemes. Due to space limitations we refer the reader to [28] for our results in the setting of chosen-ciphertext security and weak key-leakage attacks.

2 Preliminaries and Tools

In this section we present some basic notions and tools that are used in our constructions. Specifically, we present the notions of an average-case strong extractor and hash proof systems.

2.1 Randomness Extraction

The *statistical distance* between two random variables X and Y over a finite domain Ω is $\mathrm{SD}(X,Y) = \frac{1}{2}\sum_{\omega\in\Omega}|\Pr[X=\omega]-\Pr[Y=\omega]|$. We say that two variables are *ϵ-close* if their statistical distance is at most ϵ. The *min-entropy* of a random variable X is $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X=x])$.

Dodis et al. [9] formalized the notion of *average min-entropy* that captures the remaining unpredictability of a random variable X conditioned on the value of a random variable Y, formally defined as follows:

$$\widetilde{\mathrm{H}}_\infty(X|Y) = -\log\left(E_{y\leftarrow Y}\left[2^{-\mathrm{H}_\infty(X|Y=y)}\right]\right) \enspace .$$

The average min-entropy corresponds exactly to the optimal probability of guessing X, given knowledge of Y. The following bound on average min-entropy was proved in [9]:

Lemma 2.1 ([9]). *If Y has 2^r possible values and Z is any random variable, then $\widetilde{\mathrm{H}}_\infty(X|(Y,Z)) \geq \mathrm{H}_\infty(X|Z) - r$.*

A main tool in our constructions in this paper is a strong randomness extractor. The following definition naturally generalizes the standard definition of a strong extractor to the setting of average min-entropy:

Definition 2.2 ([9]). *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *is an* average-case (k,ϵ)-strong extractor *if for all pairs of random variables (X,I) such that $X \in \{0,1\}^n$ and $\widetilde{\mathrm{H}}_\infty(X|I) \geq k$ it holds that*

$$\mathrm{SD}\left((\mathsf{Ext}(X,S),S,I),(U_m,S,I)\right) \leq \epsilon \ ,$$

where S is uniform over $\{0,1\}^t$.

Dodis et al. proved that any strong extractor is in fact an average-case strong extractor, for an appropriate setting of the parameters. As a specific example, they proved the following generalized variant of the leftover hash lemma, stating that any family of pairwise independent hash functions is an average-case strong extractor:

Lemma 2.3 ([9]). *Let X,Y be random variables such that $X \in \{0,1\}^n$ and $\widetilde{\mathrm{H}}_\infty(X|Y) \geq k$. Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^m$. Then for $h \leftarrow \mathcal{H}$ and for any $m \leq k - 2\log(1/\epsilon)$, it holds that*

$$\mathrm{SD}\left((Y,h,h(X)),(Y,h,U_m)\right) \leq \epsilon.$$

2.2 Hash Proof Systems

We present the framework of hash proof systems, introduced by Cramer and Shoup [7]. For simplicity we frame the description by viewing hash proof systems as key-encapsulation mechanisms (using the notation of Kiltz et al. [22]), and refer the reader to [7] for a more complete description.

A key-encapsulation mechanism is a public-key encryption scheme that is used for encrypting random messages. Typically, these messages are used as encryption keys for a symmetric-key encryption scheme, which in turn encrypts the actual plaintext. In this setting, hash proof systems may be viewed as key-encapsulation mechanisms in which ciphertexts can be generated in two modes. Ciphertexts generated using the first mode are referred to as *valid ciphertexts*, and are indeed encapsulations of symmetric keys. That is, given a public key and a valid ciphertext, the encapsulated key is well defined, and can be decapsulated using the secret key. In addition, the generation process of a valid ciphertext also produces a "witness" to the fact that the ciphertext is indeed valid. Ciphertexts generated using the second mode are referred to as *invalid ciphertexts*, and essentially contain no information on the encapsulated key. That is, given a public key and an invalid ciphertext, the distribution of the encapsulated key (as it will be produced by the decryption process) is almost uniform. This is achieved by introducing redundancy into the secret key: each public key has many corresponding secret keys. The only computational requirement is that the two modes are computational indistinguishable: any efficient adversary that is given a public key cannot distinguish with a noticeable advantage between valid ciphertexts and invalid ciphertexts. We note that the secret and public keys are always generated using the same algorithm, and the indistinguishability requirement is only over the ciphertexts.

Smooth projective hashing. Let $\mathcal{SK}$, $\mathcal{PK}$, and $\mathcal{K}$ be sets where we view $\mathcal{SK}$ as the set of secret keys, $\mathcal{PK}$ as the set of public keys, and $\mathcal{K}$ as the set of encapsulated symmetric keys. Let $\mathcal{C}$ and $\mathcal{V} \subset \mathcal{C}$ be sets, where we view $\mathcal{C}$ as the set of all ciphertexts, $\mathcal{V}$ as the set of all valid ciphertexts (i.e., those generated appropriately with a corresponding witness). We assume that there are efficient algorithms for sampling $sk \in \mathcal{SK}$, $C \in \mathcal{V}$ together with a witness w, and $C \in \mathcal{C} \setminus \mathcal{V}$.

Let $\Lambda_{sk} : \mathcal{C} \to \mathcal{K}$ be a hash function indexed with $sk \in \mathcal{SK}$ that maps ciphertexts to symmetric keys. The hash function $\Lambda_{(\cdot)}$ is *projective* if there exists a projection $\mu : \mathcal{SK} \to \mathcal{PK}$ such that $\mu(sk) \in \mathcal{PK}$ defines the action of Λ_{sk} over the subset $\mathcal{V}$ of valid ciphertexts. That is, for every valid ciphertext $C \in \mathcal{V}$, the value $K = \Lambda_{sk}(C)$ is uniquely determined by $pk = \mu(sk)$ and C. In other words, even though there are many different secret keys sk corresponding to the same public key pk, the action of Λ_{sk} over the subset of valid ciphertexts in completely determined by the public key pk. On the other hand, the action of Λ_{sk} over the subset of invalid ciphertexts should be completely undetermined: A projective hash function is *ϵ-almost 1-universal* if for all $C \in \mathcal{C} \setminus \mathcal{V}$,

$$\mathrm{SD}\left((pk, \Lambda_{sk}(C)), (pk, K)\right) \leq \epsilon \tag{1}$$

where $sk \in \mathcal{SK}$ and $K \in \mathcal{K}$ are sampled uniformly at random, and $pk = \mu(sk)$.

Hash proof systems. A hash proof system $\mathsf{HPS} = (\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ consists of three polynomial-time algorithms. The algorithm $\mathsf{Param}(1^n)$ generates parameterized instances of the form $(\mathsf{group}, \mathcal{K}, \mathcal{C}, \mathcal{V}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu)$, where group may contain public parameters. The public evaluation algorithm Pub is used to decapsulate valid ciphertexts $C \in \mathcal{V}$ given a "witness" w of the fact that C is indeed valid (specifically, one can think of w as the random coins used to sample C from the set $\mathcal{V}$). The algorithm Pub receives as input a public key $pk = \mu(sk)$, a valid ciphertext $C \in \mathcal{V}$, and a witness w of the fact that $C \in \mathcal{V}$, and outputs the encapsulated key $K = \Lambda_{sk}(C)$. The private evaluation algorithm Priv is used to decapsulate valid ciphertexts without knowing a witness w, but by using the secret key sk. That is, the algorithm Priv receives as input a secret key $sk \in \mathcal{SK}$ and a valid ciphertext $C \in \mathcal{V}$, and outputs the encapsulated key $K = \Lambda_{sk}(C)$. We assume that μ and $\Lambda_{(\cdot)}$ are efficiently computable. We say that a hash proof system is 1-universal if for all possible outcomes of $\mathsf{Param}(1^n)$ the underlying projective hash function is $\epsilon(n)$-almost 1-universal for some negligible $\epsilon(n)$.

Subset membership problem. As a computational problem we require that the *subset membership problem* is hard in HPS, which means that for random valid ciphertext $C_0 \in \mathcal{V}$ and random invalid ciphertext $C_1 \in \mathcal{C} \setminus \mathcal{V}$, the two ciphertexts C_0 and C_1 are computationally indistinguishable. This is formally captured by defining the advantage function $\mathsf{Adv}^{\mathsf{SM}}_{\mathsf{HPS},\mathcal{A}}(n)$ of an adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{SM}}_{\mathsf{HPS},\mathcal{A}}(n) = \left|\Pr_{C_0 \leftarrow \mathcal{V}}\left[\mathcal{A}(\mathcal{C},\mathcal{V},C_0)=1\right] - \Pr_{C_1 \leftarrow \mathcal{C}\setminus\mathcal{V}}\left[\mathcal{A}(\mathcal{C},\mathcal{V},C_1)=1\right]\right| \ ,$$

where $\mathcal{C}$ and $\mathcal{V}$ are generated using $\mathsf{Param}(1^n)$.

3 Defining Key-Leakage Attacks

In this section we define the notion of a key-leakage attack, as introduced as Akavia et al. [2]. Due to space limitations we refer the reader to the longer version of our paper [28] for the extension to chosen-ciphertext attacks, the definition of a weak key-leakage attack, and a discussion on several other generalizations of this framework: noisy leakage, leakage of intermediate values from the key-generation process, keys that are generated using weak random sources, and leakage of intermediate values from the decryption process.

Informally, an encryption scheme is resilient to key-leakage attacks if it is semantically secure even when the adversary obtains sensitive leakage information. This is modeled by providing the adversary with access to a leakage oracle: the adversary can submit any function f and receive $f(SK)$, where SK is the secret key. The adversary can query the leakage oracle adaptively, with only one restriction: the sum of output lengths of all the leakage functions has to be bounded by a predetermined parameter λ.

More formally, for a public-key encryption scheme $(\mathcal{G},\mathcal{E},\mathcal{D})$ we denote by $\mathcal{SK}_n$ and $\mathcal{PK}_n$ the sets of secret keys and public keys that are produced by $\mathcal{G}(1^n)$. That is, $\mathcal{G}(1^n) : \{0,1\}^* \rightarrow \mathcal{SK}_n \times \mathcal{PK}_n$ for every $n \in \mathbb{N}$. The leakage oracle, denoted $\mathsf{Leakage}(SK)$, takes as input a function $f : \mathcal{SK}_n \rightarrow \{0,1\}^*$ and outputs $f(SK)$. We say that an oracle machine $\mathcal{A}$ is a λ-key-leakage adversary if the sum of output lengths of all the functions that $\mathcal{A}$ submits to the leakage oracle is at most λ.

Definition 3.1 (key-leakage attacks). *A public-key encryption scheme $\Pi = (\mathcal{G},\mathcal{E},\mathcal{D})$ is* semantically secure against $\lambda(n)$-key-leakage attacks *if for any probabilistic polynomial-time $\lambda(n)$-key-leakage adversary $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ it holds that*

$$\mathsf{Adv}^{\mathsf{Leakage}}_{\Pi,\mathcal{A}}(n) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{Leakage}}_{\Pi,\mathcal{A}}(0)=1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{Leakage}}_{\Pi,\mathcal{A}}(1)=1\right]\right|$$

is negligible in n, where $\mathsf{Expt}^{\mathsf{Leakage}}_{\Pi,\mathcal{A}}(b)$ is defined as follows:

1. $(SK,PK) \leftarrow \mathcal{G}(1^n)$.
2. $(M_0,M_1,\mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Leakage}(SK)}(PK)$ *such that* $|M_0| = |M_1|$.
3. $C \leftarrow \mathcal{E}_{pk}(M_b)$.
4. $b' \leftarrow \mathcal{A}_2(C,\mathsf{state})$
5. *Output* b'.

Challenge-dependent key leakage. Note that the adversary is not allowed to access the leakage oracle after the challenge phase. This restriction is necessary: the adversary can clearly encode the decryption algorithm, the challenge ciphertext, and the two messages M_0 and M_1 into a function that outputs the bit b. It will be very interesting to find an appropriate definition that allows a certain form of challenge-dependent leakage.

Adaptivity. As pointed out by Akavia et al. [2], Definition 3.1 is in fact equivalent to a definition in which the adversary queries the leakage oracle only once. Informally, the adversary can encode its adaptive behavior into a single polynomial-size leakage function. It is not clear, however, that the same equivalence holds when we extend the definition to consider chosen-ciphertext attacks. Therefore, for consistency, we chose to present this adaptive definition.

4 A Generic Construction from Hash Proof Systems

In this section we present a generic construction of a public-key encryption scheme that is resilient to key-leakage attacks. We then present an instantiation of our generic construction with a simple and efficient hash proof system based on the DDH assumption. The resulting encryption scheme is resilient to any leakage of $L(1/2 - o(1))$ bits, where L is the length of the secret key. Although one can instantiate our generic construction with any hash proof system, we find this specific instantiation rather elegant.

The construction. Let $\mathsf{HPS} = (\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ be an ϵ_1-almost 1-universal hash proof system (see Section 2.2 for an overview of hash proof systems), where $\mathsf{Param}(1^n)$ generates parameterized instances of $(\mathsf{group}, \mathcal{K}, \mathcal{C}, \mathcal{V}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu)$ which are used as the public parameters of the encryption scheme. Let $\lambda = \lambda(n)$ be a bound on the amount of leakage, and let $\mathsf{Ext} : \mathcal{K} \times \{0,1\}^t \to \{0,1\}^m$ be an average-case $(\log|\mathcal{K}| - \lambda, \epsilon_2)$-strong extractor. We assume that ϵ_1 and ϵ_2 are negligible in the security parameter. The following describes the encryption scheme $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- **Key generation:** Choose a random $sk \in \mathcal{SK}$ and let $pk = \mu(sk) \in \mathcal{PK}$. Output the pair (sk, pk).
- **Encryption:** On input a message $M \in \{0,1\}^m$, choose a random $C \in \mathcal{V}$ together with a corresponding witness w, and a random seed $s \in \{0,1\}^t$. Let $\Psi = \mathsf{Ext}(\mathsf{Pub}(pk, C, w), s) \oplus M$, and output the ciphertext (C, s, Ψ).
- **Decryption:** On input a ciphertext (C, s, Ψ), output the message $M = \Psi \oplus \mathsf{Ext}(\Lambda_{sk}(C), s)$.

The correctness of the scheme follows from the property that $\Lambda_{sk}(C) = \mathsf{Pub}(pk, C, w)$ for any $C \in \mathcal{V}$ with witness w. Thus, a decryption of an encrypted plaintext is always the original plaintext. The security of the scheme (i.e., its resilience to key leakage) follows from the universality of the proof system (see Equation (1) in Section 2.2): for all $C \in \mathcal{C} \setminus \mathcal{V}$ it holds that

$$\mathrm{SD}\left((pk, \Lambda_{sk}(C)), (pk, K)\right) \le \epsilon_1 \ ,$$

where $sk \in \mathcal{SK}$ and $K \in \mathcal{K}$ are sampled uniformly at random, and $pk = \mu(sk)$. Therefore, even given pk and any leakage of λ bits, the distribution $\Lambda_{sk}(C)$ is ϵ_1-close to a distribution with *average min-entropy* at least $\log |\mathcal{K}| - \lambda$. The strong extractor is then applied to $\Lambda_{sk}(C)$ using a fresh seed (chosen during the challenge phase and thus independent of the leakage), and guarantees that the plaintext is properly hidden. The following theorem establishes the security of the scheme:

Theorem 4.1. *Assuming that* HPS *is a* 1*-universal hash proof system, the encryption scheme Π is semantically secure against $\lambda(n)$-key-leakage attacks for any $\lambda(n) \leq \log |\mathcal{K}| - \omega(\log n) - m$, where n is the security parameter and m is the length of plaintexts.*

Example: A DDH-based instantiation. Let $\mathbb{G}$ be a group of prime order q, let $\lambda = \lambda(n)$ be the leakage parameter, and let $\mathsf{Ext} : \mathbb{G} \times \{0,1\}^t \to \{0,1\}^m$ be an average-case $(\log q - \lambda, \epsilon)$-strong extractor for some negligible $\epsilon = \epsilon(n)$.

– **Key generation:** Choose $x_1, x_2 \in \mathbb{Z}_q$ and $g_1, g_2 \in \mathbb{G}$ uniformly at random. Let $h = g_1^{x_1} g_2^{x_2}$, and output the keys

$$SK = (x_1, x_2), \quad PK = (g_1, g_2, h) \quad .$$

– **Encryption:** On input a message M, choose $r \in \mathbb{Z}_q$ and $s \in \{0,1\}^t$ uniformly at random, and output the ciphertext

$$(g_1^r, g_2^r, s, \mathsf{Ext}(h^r, s) \oplus M) \quad .$$

– **Decryption:** On input a ciphertext (u_1, u_2, s, e), output $e \oplus \mathsf{Ext}(u_1^{x_1} u_2^{x_2}, s)$.

The hash proof system underlying the above encryption scheme is a well-known DDH-based 1-universal hash proof system [7], and as an immediate consequence we obtain the following corollary of Theorem 4.1:

Corollary 4.2. *Assuming the hardness of DDH, the above encryption scheme is semantically-secure against $(L/2 - \omega(\log n) - m)$-key-leakage attacks, where n denotes the security parameter, $L = L(n)$ denotes the length of the secret key and $m = m(n)$ denotes the length of the plaintext.*

5 Improved Resilience Based on DDH and *d*-Linear

In this section we propose two encryption schemes that are resilient to any key leakage of $L(1 - o(1))$ bits, where L is the length of the secret key. These proposals are based on the observation that our generic construction from hash proof systems can in fact be based on hash proof systems with a slightly weaker 1-universality property. Specifically, the 1-universality property asks that for *all* $C \in \mathcal{C} \setminus \mathcal{V}$ it holds that

$$\mathrm{SD}\left((pk, \Lambda_{sk}(C)), (pk, K)\right) \leq \epsilon$$

where $sk \in \mathcal{SK}$ and $K \in \mathcal{K}$ are sampled uniformly at random, and $pk = \mu(sk)$. It is rather straightforward that our generic construction only requires this property to hold *with overwhelming probability* over the choice of $C \in \mathcal{C} \setminus \mathcal{V}$.

For our first proposal we construct a new hash proof system that is based on the d-Linear assumption (for any $d \geq 1$) and satisfies this weaker 1-universality property[4]. The hash proof system is a generalization of the hash proof system underlying the simple instantiation described in Section 4. The resulting encryption scheme is then obtained by instantiating our generic construction with this hash proof system.

Our second proposal is a recent encryption scheme of Boneh et al. [5], that is secure under key cycles (and more generally, under encryptions of linear functions of the secret keys). This is the first and only known encryption scheme with this property. We refer to this scheme as the BHHO scheme, and show that it fits into our generic approach using an appropriate hash proof system (that satisfies the same weaker universality property). As a corollary we derive that the BHHO scheme is resilient to any leakage of $L(1 - o(1))$ bits[5].

We then provide a comparison between these two proposed schemes, indicating that the two schemes complement each other in terms of efficiency.

5.1 Proposal 1: A New Hash Proof System

We begin by presenting the encryption scheme, and then turn to describe the underlying hash proof system and its properties.

Notation. Let $\mathbb{G} = (G, q, g)$ where G a group of order q that is generated by g. For two vectors $v = (g_1, \ldots, g_k) \in \mathbb{G}^k$ and $u = (u_1, \ldots, u_k) \in \mathbb{Z}_q^k$ we define $v \cdot u^{\mathsf{T}} = \prod_{i=1}^{k} g_i^{u_i}$, and note the notation naturally extends to matrix-vector and matrix-matrix multiplications.

The encryption scheme. Let $k = k(n) \geq d+1$ be any polynomial, let $\lambda = \lambda(n)$ be the leakage parameter, and let $\mathsf{Ext} : \mathbb{G}^{k-d} \times \{0,1\}^t \to \{0,1\}^m$ be an average-case $((k-d)\log q - \lambda, \epsilon)$-strong extractor for some negligible $\epsilon = \epsilon(n)$.

The following encryption scheme has a secret key of size essentially $k \log q$ bits (k group elements), and is resilient to any leakage of $\lambda \leq (k-d)\log q - \omega(\log n) - m$ bits, where m is the length of plaintexts. That is, the scheme is resilient to any leakage of essentially a $(1 - d/k)$-fraction of the length of the secret key.

– **Key generation:** Choose $x \in \mathbb{Z}_q^k$ and $\Phi \in \mathbb{G}^{d \times k}$ uniformly at random. Let $y = \Phi x \in \mathbb{G}^d$, and output the keys

$$SK = x, \quad PK = (\Phi, y) \ .$$

– **Encryption:** On input a message M, choose $R \in \mathbb{Z}_q^{(k-d)\times d}$ and $s \in \{0,1\}^t$ uniformly at random, and output the ciphertext

$$(R\Phi, s, \mathsf{Ext}\left(Ry, s\right) \oplus M) \ .$$

[4] Recall that the DDH is the 1-Linear assumption.

[5] We note that not every circular-secure scheme is also resilient to key leakage.

– **Decryption:** On input a ciphertext (Ψ, s, e) output $e \oplus \mathsf{Ext}(\Psi x, s)$.

The following theorem establishes the security of the scheme:

Theorem 5.1. *Assuming the hardness of d-Linear, for any polynomial $k = k(n) \geq d+1$ the above encryption scheme is semantically-secure against a $((1 - d/k)L - \omega(\log n) - m)$-key-leakage attack, where n denotes the security parameter, $L = L(n)$ denotes the length of the secret key and $m = m(n)$ denotes the length of the plaintext.*

The hash proof system. Let $k = k(n) \geq d+1$ be any polynomial, and let $\mathsf{Ext} : \mathbb{G}^{k-d} \times \{0,1\}^t \to \{0,1\}^m$ be a $((k-d)\log q, \epsilon)$-strong extractor for some negligible $\epsilon = \epsilon(n)$.

We define a hash proof system $\mathsf{HPS} = (\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ as follows. The algorithm $\mathsf{Param}(1^n)$ generates instances $(\mathsf{group}, \mathcal{K}, \mathcal{C}, \mathcal{V}, \mathcal{SK}, \mathcal{PK}, \Lambda, \mu)$, where:

– $\mathsf{group} = (\mathbb{G}, \Phi, s)$, where $\Phi \in \mathbb{G}^{d \times k}$ and $s \in \{0,1\}^t$ are chosen uniformly at random.
– $\mathcal{C} = \mathbb{G}^{(k-d)\times k}$, $\mathcal{V} = \left\{ R\Phi \ : \ R \in \mathbb{Z}_q^{(k-d)\times d} \right\}$, $\mathcal{K} = \{0,1\}^m$.
– $\mathcal{SK} = \mathbb{Z}_q^k$, $\mathcal{PK} = \mathbb{G}^d$.
– For $sk = x \in \mathcal{SK}$ we define $\mu(sk) = \Phi x \in \mathcal{PK}$.
– For $C \in \mathcal{V}$ with witness $R \in \mathbb{Z}_q^{(k-d)\times d}$ we define $\mathsf{Pub}(pk, C, R) = \mathsf{Ext}(Ry, s)$.
– For $C \in \mathcal{V}$ we define $\mathsf{Priv}(sk, C) = \Lambda_{sk}(C) = \mathsf{Ext}(Cx, s)$.

5.2 Proposal 2: The BHHO Scheme

We show that a simple setting of the parameters in the BHHO encryption scheme [5] results in an encryption scheme that is resilient any key leakage of $L(1 - o(1))$ bits, where L is the length of the secret key. Let $\mathbb{G} = (G, q, g)$ where G a group of order q that is generated by g, and set $\ell = \lambda + 2\log q + 2\log(1/\epsilon)$ for some negligible $\epsilon = \epsilon(n)$.

– **Key generation:** Choose $s_1, \ldots, s_\ell \in \{0,1\}$ and $g_1, \ldots, g_\ell \in \mathbb{G}$ uniformly at random. Let $h = \prod_{i=1}^{\ell} g_i^{s_i}$, and output the keys

$$SK = (s_1, \ldots, s_\ell), \quad PK = (g_1, \ldots, g_\ell, h) \ .$$

– **Encryption:** On input a message $M \in G$, choose $r \in \mathbb{Z}_q$ uniformly at random, and output the ciphertext

$$(g_1^r, \ldots, g_\ell^r, h^r \cdot M) \ .$$

– **Decryption:** On input a ciphertext $(u_1, \ldots, u_k, e)$ output $e \cdot \left(\prod_{i=1}^{\ell} u_i^{s_i} \right)^{-1}$.

The encryption scheme can be viewed as based on a hash proof system with the following subset membership problem (whose hardness follows from DDH):

$$\begin{aligned} \mathcal{C} &= \{(g_1^{r_1}, \ldots, g_\ell^{r_\ell}) : r_1, \ldots, r_\ell \in \mathbb{Z}_q\} \\ \mathcal{V} &= \{(g_1^{r}, \ldots, g_\ell^{r}) : r \in \mathbb{Z}_q\} \ . \end{aligned}$$

The leftover hash lemma guarantees that with overwhelming probability over the choice of $C = (u_1, \dots, u_\ell) \in \mathcal{C} \setminus \mathcal{V}$ it holds that $\Lambda_{sk}(C) = \prod_{i=1}^{\ell} u_i^{s_i}$ is ϵ-close to the uniform distribution over G, even given $h = \prod_{i=1}^{\ell} g_i^{s_i}$ and any leakage of length λ bits.

5.3 Comparison

The main difference between the two schemes proposed in this section is in their method of extracting randomness from the secret key. In the first proposal an invertible function is applied to the secret key (thus preserving its min-entropy), and then a strong extractor is applied to the resulting value. In the second proposal, the entropy of the secret key is extracted directly using the subset-product hash functions. Although the second proposal uses a more direct method to extract the randomness of the secret key, it requires the subset-product functions to operate on the *individual bits* of the secret key, since otherwise the subset-product functions are not pairwise independent. In contrast, in the first proposal the extractor operates on the secret key as *group elements*. This leads to significant differences in performance.

For any leakage parameter λ, the size of the secret keys in the two proposals is essentially the same, whereas the size of the public key in the first proposal is shorter by a factor of $\log q$. When considering the length of the ciphertexts and the number of exponentiations per ciphertext, the first proposal performs better than the second proposal when roughly $\lambda < L(1 - 1/\log q)$, where L is the length of the secret key (note that such a λ is a considerable amount of leakage). For example, by setting $k = 2$ in the first proposal one obtains the simple instantiation described in Section 4 which is resilient to any leakage of $L(1/2 - o(1))$ bits, and requires only 3 exponentiations per ciphertext. For achieving the same resilience in the second proposal more than $\log q$ exponentiations are required. In Table 1 we present a comparison between the efficiency of the schemes. Since the second proposal scheme is based on DDH and encrypts group elements, we compare it to the first proposal using $d = 1$ (i.e., based on DDH), and $m = \log q$ (i.e., $\log q$-bit plaintexts). For simplicity we assume that λ is a multiple of $\log q$, and note that the table presents asymptotical estimates, and not exact numbers.

Table 1. Comparison between the two proposals

	Proposal 1 (Section 5.1)	Proposal 2 (Section 5.2)
Secret key (bits)	$\lambda + 2\log q + 2\log(1/\epsilon)$	$\lambda + 2\log q + 2\log(1/\epsilon)$
Public key (bits)	$\lambda + 2\log q + 2\log(1/\epsilon)$	$\log q\,(\lambda + 2\log q + 2\log(1/\epsilon))$
Ciphertext (bits)	$\frac{(\lambda+2\log q+2\log(1/\epsilon))^2}{\log q}$	$\log q\,(\lambda + 2\log q + 2\log(1/\epsilon))$
Exponentiations	$\left(\frac{\lambda+2\log q+2\log(1/\epsilon)}{\log q}\right)^2$	$\lambda + 2\log q + 2\log(1/\epsilon)$

6 Generalized Forms of Key-Leakage Attacks

In this section we present several generalizations of the framework considered in this paper that are satisfied by our schemes. Due to space limitations we describe these generalizations very briefly and refer the reader to [28] for more details.

Noisy leakage. In the side-channel attack of Halderman et al. [18] the adversary learns a noisy version of all of the memory. This is a more general scenario than the scenario captured by Definition 3.1: The leakage is not of bounded length, but it is guaranteed that the secret key is still somewhat unpredictable given the leakage. In the information-theoretic setting, this generalization does not necessarily strengthen the definition, since the leakage may be compressed to essentially λ bits. However, in the computational setting (which is the setting we consider in this work) we can conjecture that this notion is stronger.

Leakage of intermediate values from the key-generation process. Definition 3.1 assumes that the adversary does not learn any of the intermediate values that occur during the generation of the secret and public keys. In practice, however, this is not always a valid assumption. Specifically, in the attack of Halderman et al. [18] the adversary learns a noisy version of all of the memory, and it is rather likely that intermediate values from the generation of the keys are not always completely erased. This motivates a natural generalization that allows the adversary to learn functions of the random bits that are used by the key generation algorithm. Encryption schemes that satisfy this notion of security are more robust to leakage in the sense that the key generation algorithm does not have to make sure that all intermediate key-related values have been deleted. In addition, this generalization is especially important to security under composition of cryptographic primitives. For example, the key generation algorithm may use random bits (or pseudorandom bits) that are the output of another primitive (say, a pseudorandom generator) which may also suffer from unintended leakage of sensitive information.

Keys generated using weak random sources. When considering leakage of the random bits that are used by the key generation algorithm, then from the adversary's point of view these bits are uniformly distributed subject to the leakage information. A natural generalization is to consider cases in which the keys are generated using a weak source of random bits. This is relevant, in particular, in light of crucial security vulnerabilities that were recently identified in pseudorandom generators that are used by many systems [13,17,36].

Leakage of intermediate values from the decryption process. An additional generalization is to consider leakage that may occur during computation, and not only leakage from the stored key. Specifically, an invocation of the decryption algorithm may produce various *intermediate* values, whose leakage may compromise the security of the scheme even if the scheme is robust against leakage from the stored key. Such a notion of security is generically guaranteed when considering leakage of bounded length. However, it is not always guaranteed when the adversary obtains all of the memory in a noisy fashion.

Consider the seemingly contrived example of a decryption algorithm that first encodes the secret key using a good error-correcting code, and then performs the actual decryption. In this case, an adversary that obtains a noisy variant of the memory can clearly recover the secret key. This example, however, is not so contrived, since as demonstrated by Halderman et al., encryption schemes typically compute intermediate key-related values whose representation is rather redundant, and this can be used to attack the scheme. Moreover, even if the encryption scheme itself does not explicitly instructs to compute intermediate values, it may be the case that such values are computed by a specific implementation of the encryption scheme.

References

1. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC, pp. 474–495 (2009)
3. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
5. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008)
6. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
7. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
8. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. 33(1), 167–226 (2003)
9. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
10. Dodis, Y., Sahai, A., Smith, A.: On perfect and adaptive security in exposure-resilient cryptography. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 301–324. Springer, Heidelberg (2001)
11. Dodis, Y., Tauman Kalai, Y., Lovett, S.: On cryptography with auxiliary input. To appear in STOC (2009)
12. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM J. Comput. 30(2), 391–437 (2000)
13. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the windows random number generator. In: ACM CCS, pp. 476–485 (2007)
14. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302 (2008)

15. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
16. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
17. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the linux random number generator. In: IEEE Symposium on Security and Privacy, pp. 371–385 (2006)
18. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX, pp. 45–60 (2008)
19. Heninger, N., Shacham, H.: Improved RSA private key reconstruction for cold boot attacks. Cryptology ePrint Archive, Report 2008/510 (2008)
20. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
21. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
22. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: EUROCRYPT, pp. 590–609 (2009)
23. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
24. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
25. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
26. Maurer, U.M.: On the oracle complexity of factoring integers. Computational Complexity 5(3-4), 237–247 (1995)
27. Micali, S., Reyzin, L.: Physically observable cryptography. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
28. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. Cryptology ePrint Archive, Report 2009/105 (2009)
29. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437 (1990)
30. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: ASIACCS, pp. 56–65 (2008)
31. Pietrzak, K.: A leakage-resilient mode of operation. In: EUROCRYPT, pp. 462–482 (2009)
32. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)
33. Rivest, R.L., Shamir, A.: Efficient factoring based on partial information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986)
34. Shacham, H.: A Cramer-Shoup encryption scheme from the Linear assumption and from progressively weaker Linear variants. Cryptology ePrint Archive, Report 2007/074 (2007)
35. Tauman Kalai, Y., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs and applications (2009)
36. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S.: PRNG PR0N: Understanding the Debian OpenSSL debacle (2008)

Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model

Joël Alwen, Yevgeniy Dodis, and Daniel Wichs

Department of Computer Science, New York University
{jalwen,dodis,wichs}@cs.nyu.edu

Abstract. We study the design of cryptographic primitives resilient to key-leakage attacks, where an attacker can repeatedly and adaptively learn information about the secret key, subject *only* to the constraint that the *overall amount* of such information is bounded by some parameter ℓ. We construct a variety of leakage-resilient public-key systems including the first known identification schemes (ID), signature schemes and authenticated key agreement protocols (AKA). Our main result is an efficient three-round AKA in the Random-Oracle Model, which is resilient to key-leakage attacks that can occur prior-to *and* after a protocol execution. Our AKA protocol can be used as an interactive encryption scheme with qualitatively stronger privacy guarantees than non-interactive encryption schemes (constructed in prior and concurrent works), which are inherently insecure if the adversary can perform leakage attacks after seing a ciphertext.

Moreover, our schemes can be flexibly extended to the *Bounded-Retrieval Model*, allowing us to tolerate very large absolute amount of adversarial leakage ℓ (potentially many gigabytes of information), *only* by increasing the size of the secret key and without any other loss of efficiency in communication or computation. Concretely, given any leakage parameter ℓ, security parameter λ, and any desired fraction $0 < \delta \leq 1$, our schemes have the following properties:

– Secret key size is $\ell(1 + \delta) + O(\lambda)$.
– Public key size is $O(\lambda)$, and independent of ℓ.
– Communication complexity is $O(\lambda/\delta)$, and independent of ℓ.
– Computation reads $O(\lambda/\delta^2)$ locations of the secret key, independent of ℓ.

Lastly, we show that our schemes allow for repeated *"invisible updates"* of the secret key, allowing us to tolerate up to ℓ bits of leakage in between any two updates, and an unlimited amount of leakage overall. These updates require that the parties can securely store a short "master update key" (e.g. on a separate secure device protected against leakage), which is only used for updates and not during protocol execution. The updates are invisible in the sense that a party can update its secret key at any point in time, *without* modifying the public key or notifying the other users.

1 Introduction

Traditionally, cryptographic systems rely on complete privacy of cryptographic keys. Unfortunately, this idealized assumption is often hard to satisfy in real systems. In many situations, the attacker might get some partial information about secret keys through means which were not anticipated by the designer of the system and, correspondingly, not taken into account when arguing its security. Such attacks, referred to as *key-leakage attacks*, come in a large variety. For example, this includes *side-channel* attacks [20,21,28], where an adversary observes some "physical output" of a computation (radiation, power, temperature, running time etc.) in addition to the "logical output" of the computation. Alternatively, this also includes the "cold-boot" attack of Halderman et al. [15], where an adversary can learn (imperfect) information about memory contents, even after a machine is powered down. Lastly, this can include various malware/virus/hacking attacks where the adversary can download arbitrary information from an attacked computer.

Given that one cannot hope to eliminate the problem of leakage attacks altogether, it is natural to design leakage-resilient cryptographic schemes which remain (provably) secure, even in the face of such attacks. To do so, we must first decide on an appropriate model of what information the adversary can learn during a leakage attack. In this work, we assume that the attacker can repeatedly and adaptively learn *arbitrary functions* of the secret key sk, as long as the *total* number of bits leaked during the lifetime of the system is bounded by some parameter ℓ. Due to its generality, this model seems to include a very large class of attacks mentioned above, and has recently attracted a lot of attention from the research community. In particular, this model simultaneously covers the following two typical scenarios, which seem to be treated differently in the existing literature.

Relative Leakage. Here, the secret key is chosen to be of some particular length s, which depends on the security parameter, and we assume that the leakage ℓ is bounded by some shrinking function of s; e.g., the attacker's leakage is less than half of the key-size. This assumption seems to be natural for modeling attacks where, no matter what the key-size is, the attacker gets some imperfect reading of the key. For example, this naturally models "cold boot attacks" attacks [15] (where the attacker might get part of the key stored in RAM) and "microwave" attacks (where the attacker manages to extract a corrupted copy of the key from a smart-card), among others.

Bounded-Retrieval Model (BRM). Here we assume that there is an external natural bound ℓ on the overall amount of information the attacker can learn throughout the lifetime of the system, particularly concentrating on the setting when ℓ can be extremely large. For example, the attacker may be able to repeatedly perform many side-channel attacks, each of which reveals a few bits of information about the key but, if the bandwidth of such attacks is relatively small, it may be infeasible, too time consuming, or simply not cost-affective for the adversary to learn "too much" information (say, more than 10 megabytes) overall. Alternatively, if an attacker hacks into a remote system (or infects it with some malware) it may again be infeasible/impractical for the attacker to download "too much" data (say, more than 10 gigabytes). In these situations the leakage bound ℓ is decided by external factors and one can only resist such attacks by making the secret key *intentionally large*, to dominate ℓ. Therefore, we want to be able to set the key size flexibly depending on the security parameter *and* the leakage bound ℓ. By itself, having large secret-keys might not be a big problem for usability, given the extremely cheap price of storage nowadays. Therefore, the main goal of this setting, usually refereed to as the *Bounded-Retrieval Model* (BRM) [8,12], is to ensure that the *necessary* inefficiency in storage is essentially the *only* inefficiency that the users of the system incur. In particular, honest users should only have to read a small portion of the secret (this is called *locality*), and the public keys, communication and computation should not be much larger than in conventional cryptosystems. In particular, all efficiency parameters other than the secret-key size should *only* be proportional to the security parameter, and *not* the leakage bound ℓ.

To summarize, both leakage models (relative and BRM) study essentially the same technical question. However, the BRM setting additionally demands that: *users can increase their secret key size flexibly, so as to allow for an arbitrary large leakage bounds ℓ, but without degrading other efficiency parameters, such as computation, communication and locality.* This is the perspective we will take in this paper, treating both settings together, while striving to allow for the flexibility of the BRM.

NOTIONS OF SECURITY. Security with respect to key leakage attacks can be defined for nearly all cryptographic primitives (e.g. encryption, signatures, authenticated key agreement . . .) However, for many of the above primitives, there are natural limitations on the security notions that can be achieved in the presence of such attacks. For example, encryption schemes lose all privacy if the adversary can perform leakage attacks *after* seeing a ciphertext, since the leakage function can simply decrypt it and reveal some information about plaintext. Similarly, one cannot achieve *existential unforgeability* for signature schemes if the leakage bound ℓ is larger than the size of a single signature (as is the case in the BRM), since the adversary can simply leak the signature of a message of her choosing. These limitations do not seem to apply when considering interactive primitives, and therefore we choose to concentrate on *authenticated key agreement (AKA)*, which in turn allows for interactive encryption and authentication, and achieves *qualitatively* stronger security guarantees, even in the BRM.

1.1 Our Results

Our main result is the construction of a leakage-resilient public-key authenticated key agreement (AKA) protocol with the flexibility required by the BRM. We assume a public-key infrastructure where users have short public-keys and flexibly sized (potentially huge) secret keys. In a leakage-resilient AKA protocol, pairs of users agree on shared session-keys which are private and authentic, even if: (a) the attacker learns at most ℓ bits of information about the secret keys of both users *prior to* the protocol execution; (b) the attacker may learn the secret keys entirely *after* the protocol execution. In particular, condition (a) ensures that the adversary cannot *impersonate* an honest user, even after learning ℓ bits of leakage about that user's secret key. Since the shared session-keys can safely be used for encryption/authentication, a public-key AKA naturally yields *interactive* public-key encryption and authentication schemes which are secure under assumptions (a) and (b), and do not suffer from the inherent limitations of their non-interactive counterparts.

ROADMAP OF AKA CONSTRUCTION. Our construction of AKA is based on simpler primitives and, in particular, we also construct identification schemes (ID) and (non-interactive) signature schemes, which are of interest in their own right. The main technical portion of our paper will be the construction of ID schemes secure against leakage attacks. We then apply the Fiat-Shamir heuristic to obtain efficient leakage-resilient signature schemes in the random oracle (RO) model. Of course, our signature schemes cannot provide *existential unforgeability*, if the allowed leakage exceeds the size of even a single signature (which is usually the case in the BRM). Interestingly, we show how to achieve existential unforgeability under this necessary constraint, which resolves an open problem mentioned in [1]. For the BRM setting, which is our main point of interest, we must settle for a weaker, but still very useful security notion, that we call *entropic unforgeability*. Although an attacker may be able to forge signatures for a few messages after she performs a key-leakage attack, she should be unable to forge the signature of a random message sampled from any distribution of high enough min-entropy.

Finally, we use a standard construction of AKA based on Diffie-Hellman key exchange, in which the parties bind the protocol execution to a particular session and to their identities using signatures. We plug our entropically secure signature scheme into this construction to get leakage-resilient AKA. Intuitively, the usage of entropically secure signature will suffice, since each party only signs messages which are partially controlled by the other party, and happen to have entropy. We note that our constructions of authenticated key agreement from entropic signatures, and our constructions of such signatures from ID schemes, are extremely efficient and essentially preserve (1) the long-term public/secret key size, (2) the communication complexity, (3) the locality, and (4) the allowed leakage. Therefore, we apply most of our efforts to the construction of optimized, leakage-resilient ID schemes.

ID SCHEME CONSTRUCTIONS. We present three ID scheme constructions, which build on top of one another. First we notice that a generalization of the discrete-log based Okamoto ID scheme [25] using m generators, denoted $\mathrm{Okamoto}_m^\lambda$, is secure against leakage attacks where the allowed leakage is $\ell \approx (1 - \frac{1}{m})|\mathsf{sk}|$, and can be set arbitrarily close to the size of the secret key. Our argument relies on the following three simple properties of the scheme:

(1) Any adversarial prover that impersonates some identity must *know* a corresponding secret key for the identity's public key.
(2) For any Okamoto public key, there are (exponentially) many corresponding secret keys. Moreover, the actual secret key of the scheme maintains a high level of *information-theoretic entropy*, even when given: (a) the public key, (b) protocol executions between adversarial verifier and honest prover, and (c) ℓ bits of secret-key leakage.
(3) It is computationally infeasible to come up with an Okamoto public key and two *different* corresponding secret keys.

By property (1), an adversarial prover that successfully mounts impersonation attacks knows *some* secret key for the honest user's public key and, by property (2), this secret key is (information theoretically) unlikely to match the one possessed by the honest user, *even if the adversary got ℓ bits leakage*. Therefore, the adversarial prover together with the honest user can generate two different secret keys for a single Okamoto public

Table 1. Efficiency vs. Leakage Trafeoffs For Our ID, Sig, AKA schemes

Scheme	pub. params	pk	sk	help	Comm.	Loc.	Leakage ℓ (in bits)
$\text{Okamoto}^{\lambda}_{m}$	m	1	m	0	$m + O(1)$	m	$(1-\delta)\lvert\mathsf{sk}\rvert$ $\delta \approx \frac{1}{m}$
$\text{DirProd}^{\lambda}_{n,m,t}$	m	1	nm	n	$O(tm)$	tm	$(1-\delta)\lvert\mathsf{sk}\rvert$ $\delta \approx \left(\frac{1}{m} + O\left(\frac{\lambda}{t}\right)\right)$
$\text{CompDirProd}^{\lambda}_{n,m,t}$	m	1	nm	n	$m + O(1)$	tm	$(1-\delta)\lvert\mathsf{sk}\rvert$ $\delta \approx \left(\frac{1}{m} + O\left(\frac{\lambda}{t}\right)\right)$

key, contradicting property (3) and hence proving security. We note that several other identification schemes (e.g. an alternate construction by Okamoto [25] based on RSA, and the Ong-Schnorr [26] scheme based on factoring) also have the three mentioned properties and are therefore leakage-resilient as well.

While the (generalized) Okamoto scheme already provides an adequate solution for *relative* leakage, it cannot achieve large *absolute* leakage, without a proportional increase in communication complexity and locality. Therefore, we present two extensions of the Okamoto scheme which are suitable to the BRM setting. The first extension, denoted $\text{DirProd}^{\lambda}_{n,m,t}$, is based on the idea of taking a "direct-product" of n basic Okamoto schemes, where the verifier will selects a small random subset of $t \ll n$ of these schemes, and executes the basic protocol for them in parallel. One can think of this as a simple form of *"leakage amplification"*, where we amplify the amount of allowed absolute leakage. Lastly, we improve the communication complexity of this second scheme still further (in the Random Oracle model), by showing how to use ideas from coding-theory and the special structure of the Okamoto scheme, to "securely compress" the t chosen Okamoto public keys into a *single* public key, and then running a *single* ID protocol for this key. Therefore, and quite remarkably, our third scheme, denoted $\text{CompDirProd}^{\lambda}_{n,m,t}$, has essentially the same communication complexity as the basic (non-BRM) Okamoto scheme even though the allowed leakage ℓ can be made arbitrarily large.

OVERVIEW OF ACHIEVED PARAMETERS. We summarize the main parameters of the three ID scheme constructions (which translate into essentially the same parameters for the corresponding signatures and AKA protocols) in Table 1. The columns indicate the sizes of: the public parameters shared by all users, the public key, the secret key, a helper key (which is stored locally by each user, but does not have to be kept secret), the communication complexity per party (or signature size), the locality, and the allowed leakage ℓ. For simplicity, only the leakage parameter ℓ is measured in bits, and all other quantities are measures in group elements. The parameters m, n, t offer *flexibility* to meet the various desired settings of of absolute leakage ℓ and relative leakage $(1 - \delta)$. In particular:

– For the first scheme ($\text{Okamoto}^{\lambda}_{m}$), the only flexibility is in the number of generators m. Essentially to allow for relative leakage $(1 - \delta)$ we can set $m \approx 1/\delta$ which gives us very practical schemes for reasonable settings of the relative leakage (e.g. $\delta = \frac{1}{2}$). However, to allow for a large absolute leakage ℓ, we must increase

m still further (and proportionally with ℓ), which increases the communication, computation and size of public parameters to unreasonable levels.
- For the second and third scheme ($\mathrm{DirProd}^{\lambda}_{n,m,t}$, $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$), we have the additional flexibility offered by parameters n (the number of stored copies of Okamoto key pairs) and t (the number of Okamoto keys used during a particular protocol). We notice that, by setting $m \approx 1/\delta, t \approx O(\lambda/\delta)$ we allow a relative leakage of $(1-\delta)$ and still get practical schemes with small public parameters, public key size, communication (especially in the third scheme), and locality. Moreover, we can then flexibly accommodate *any* value of the absolute leakage ℓ *only* by increasing n which *only* affects the size of the secret key.

INVISIBLE KEY UPDATES. Lastly, we mention a simple but powerful feature of our schemes. We introduce a method for users to periodically *update* their secret keys, so that the scheme remains secure as long as the adversary learns at most $\ell \approx (1-\delta)|\mathsf{sk}|$ bits of key leakage *in between updates*, but may learn leak significantly more than the size of the secret key *overall*. Our updates are *invisible* to the outside world, in the sense that the public keys remain unchanged and users do not need to know when or how often the secret keys of other users are updated in order to run an AKA protocol. For such updates, we require the use of a *"master update key"* which must be stored securely on an external storage device that is not susceptible to leakage attacks.

1.2 Related Work

WEAK SECRETS, SIDE-CHANNEL ATTACKS AND BRM. The model of key leakage attacks, as studied in this work, is very related to the study of cryptography with *weak secrets*. A weak secret is one which comes from some arbitrary distribution that has a sufficient level of (min-)entropy, and one can think of a secret key that has been partially compromised by leakage attacks as coming from such a distribution. Most of the prior work concerning weak secrets is specific to the *symmetric key setting* and much of this work is *information-theoretic in nature*. For example, the study of privacy-amplification [3,22] shows how two users, who *share* a weak secret, can agree on a uniformly random key in the presence of a passive attacker. Such information-theoretically secure schemes can only be used *once* to convert a shared symmetric-key, which may have been partially compromised by leakage attacks, into a *single* uniform session-key.

In the computational symmetric-key setting, users can agree on *arbitrarily many* session-keys using Password Authenticated Key Agreement (PAKE) [2], where they use their shared weak (or partially compromised) secret key as the password. However, these solutions do not scale to the BRM, as they do not preserve low locality when the secret is large. The Bounded-Retrieval Model (BRM), where users have a huge secret key which is subject to large amounts of adversarial leakage, was introduced by [8,12]. In particular, Dziembowski [12] constructed a *symmetric key* authenticated-key-agreement protocol for this setting in the Random-Oracle model. This was later extended to the standard model by [7]. We also note that *non-interactive* symmetric-key encryption schemes from weakly-secret keys were constructed implicitly in [27] (based on weak PRFs) and explicitly in [9] based on "learning parity with noise".

The only related prior work that considers leakage attacks in the *public-key* setting is a recent work of Akavia et al. [1], which showed that Regev's public-key encryption scheme [29] (based on lattices) is leakage-resilient. Several recent concurrent and independent works [18,19,24] also study leakage-resilient public-key primitives. The works of [18,24] present several new constructions of of leakage-resilient public-key encryption schemes for this setting, based on more general (and non-lattice) assumptions, tolerating more leakage, achieving CCA-2 security and allowing for stronger "auxiliary-input" attacks (described subsequently). We note that all such non-interactive encryption schemes inherently become insecure if the adversary can perform leakage attacks *after* seeing a ciphertext. Existentially unforgeable leakage-resilient signatures were studied in the concurrent work of Katz [19], who independently discovered our $\mathsf{Okamoto}^{\lambda}_{m}$ construction (described above) of signatures in the Random Oracle model, as well as an alternative (albeit not practically efficient) instantiation of such signatures in the standard model. None of the prior or concurrent works in the public-key setting extend to the Bounded Retrieval Model.

OTHER MODELS OF ADVERSARIAL KEY COMPROMISE. It is worth describing several related models for key compromise which differ from the one used in this work. One possibility is to restrict the *type* of information that the adversary can learn about the secret key. For example, a line of work called *exposure resilient cryptography* [5,11] studies a restricted class of adversarial leakage functions, where the adversary gets a *subset of the bits* of the secret key. In this setting, one can secure keys against leakage generically, by encoding them using an *all-or-nothing transform (AONT)*. We note that many natural side-channel attacks (e.g. learning the hamming weight of the key) and most malware attacks are not captured by this model.

Another line of work, initiated by Micali and Reyzin [23] and studied further by Dziembowski and Pietrzak [13,27], designs various symmetric-key primitives under the axiom that "only computation leaks information". In these works, each stage of computation is assumed to leak some arbitrary shrinking function of (only) the data it accesses, but the adversary can observe computation *continuously*, and can learn an unbounded amount of such information overall. In particular, this model can protect against an adversary that continuously perform side-channel attacks (such as DPA attacks), each of which leaks some partial information (only) about the "current" computation. On the other hand, the axiom that "only computation leaks information" does not seem to apply to many other natural attacks, such as the memory/microwave attacks or virtually all malware/virus attacks. A related model, where the adversary can learn the values on some subset of wires during the evaluation of a circuit, was studied by Ishai et al. [17].

Lastly, the recent works [9,18] study *auxiliary input*, where the adversary can learn functions $f(\mathsf{sk})$ of the secret key sk subject only to the constraint that such a function is *hard to invert*. This is a strictly stronger model than the one considered in this work, as such functions f can have output length larger than the size of the secret key and can reveal *all* of the statistical entropy of the key.

2 Preliminaries

ENTROPY AND PREDICTABILITY. We review the information-theoretic definition for entropy, along a new generalization useful for our paper.

Definition 1. *The* min-entropy *of a r. v.* X *is* $\mathbf{H}_\infty(X) \overset{def}{=} -\log(\max_x \Pr[X = x])$.

We can rephrase the above definition in terms of predictors $\mathcal{A}$. The min-entropy of a random variable X measures how well X can be predicted by the *best* predictor $\mathcal{A}$, i.e. $\mathbf{H}_\infty(X) = -\log(\max_{\mathcal{A}} \Pr[\mathcal{A}() = X])$, where the max is taken over all predictors without any requirement on efficiency. The work of [10], offered a natural generalization of min-entropy, called the *(average) conditional min-entropy of X conditioned on Z*, which can be defined as measuring the maximum predictability of X by a predictor that is given the value Z. In this paper, we generalize the notion of conditional min-entropy still further, to *interactive* predictors $\mathcal{A}$, which participate in some *randomized experiment* $\mathcal{E}$. We model experiments as interactions between $\mathcal{A}$ and a *challenger oracle* $\mathcal{E}(\cdot)$ which can be randomized, stateful and interactive. We consider the predictability of X by an arbitrary predictor $\mathcal{A}^{\mathcal{E}(\cdot)}$.

Definition 2. *The* conditional min-entropy *of a random variable X, conditioned on the experiment $\mathcal{E}$ is* $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}) \overset{def}{=} -\log(\max_{\mathcal{A}} \Pr[\mathcal{A}^{\mathcal{E}(\cdot)}() = X])$. *In the special case that $\mathcal{E}$ is a non-interactive experiment which simply outputs a random variable Z, we abuse notation and write $\widetilde{\mathbf{H}}_\infty(X \mid Z)$ to denote $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E})$.*

REVIEW OF Σ-PROTOCOLS. Let $\mathcal{R}$ be a relation consisting of *instance, witness* pairs $(x, w) \in \mathcal{R}$ and let $L_R = \{x \mid \exists w, (x, w) \in \mathcal{R}\}$ be the *language* of $\mathcal{R}$. A Σ-protocol for $\mathcal{R}$ is a protocol between a PPT ITM prover $\mathcal{P}(x, w)$ and a PPT ITM verifier $\mathcal{V}(x)$, which proceeds in three rounds with conversations (a, c, z) initiated by the prover. We require that a Σ-protocol satisfies *perfect completeness*, *special soundness*, and *Honest Verifier Zero Knowledge*. In the full version of the paper, we prove the following lemma.

Lemma 1. *Let $(\mathcal{P}, \mathcal{V})$ be an* HVZK *protocol for the relation $\mathcal{R}$, and let (X, W) be random variables over $\mathcal{R}$. Let $\mathcal{E}_1$ be an arbitrary experiment in which $\mathcal{A}$ is given X at the start of the experiment, and let $\mathcal{E}_2$ be the same as $\mathcal{E}_1$, except that $\mathcal{A}$ is also given oracle access to $\mathcal{P}(X, W)$ throughout the experiment. Then $\widetilde{\mathbf{H}}_\infty(W|\mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(W|\mathcal{E}_1)$.*

PRIME-ORDERED GROUPS. We use the notation $\mathcal{G}(1^\lambda)$ to denote a group sampling algorithm which, on input 1^λ, outputs a tuple $\mathbb{G} = (p, G, g)$ where p is a prime, G is a (description of a) group of order p, and g is a generator of G. We will rely on the usual hardness assumptions: the discrete-logarithm (DL), computational Diffie-Hellman assumption (CDH) and decisional Diffie-Hellman (DDH) assumptions. We will also rely on the Gap Diffie-Hellman (GDH) assumption which state that for some groups, in which the DDH problem can be solved efficiently (for example using a bilinear map), the CDH problem is still hard.

3 Leakage Oracle

We model adversarial leakage attacks on a secret key sk, by giving the adversary access to a *leakage oracle*, which the adversary can (periodically) query to gain information about sk. This oracle is defined as follows.

Definition 3. *A* leakage oracle $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$ *is parameterized by a secret key* sk*, a* leakage parameter ℓ *and a security parameter* λ*. A query to the oracle consists of a (description of)* leakage function $h_i : \{0,1\}^* \rightarrow \{0,1\}^{\alpha_i}$*. The oracle* $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$ *checks if the sum of* α_i*, over all queries received so far, exceeds the leakage parameter* ℓ *and ignores the query if this is the case. Otherwise, the oracle computes the function* $h_i(\mathsf{sk})$ *for at most* $\mathsf{poly}(\lambda)$ *steps. If the computation completes, the oracle responds with the output and, otherwise, it responds with the dummy value* 1^{α_i}*.*

Since the cumulative output of leakage-oracle queries can be guessed with probability at least $2^{-\ell}$, the oracle can decrease the entropy of sk by at most ℓ bits in any experiment.

Lemma 2. *For any random variable* SK*, any experiment* $\mathcal{E}_1$*, let* $\mathcal{E}_2$ *be the experiment which is the same as* $\mathcal{E}_1$*, but also gives the predictor access to the leakage oracle* $\mathcal{O}^{\lambda,\ell}_{\mathsf{SK}}(\cdot)$*. Then* $\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_2\ ) \geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) - \ell$*.*

4 Identification Schemes

4.1 Definition

In an identification scheme, a prover attempts to prove its identity to a verifier. This proof should be convincing and non-transferable. More formally, an identification scheme consists of the four PPT algorithms $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$:

$\mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)$**:** Outputs the public parameters of the scheme, which are common to all users. These parameters are available as inputs to $\mathsf{KeyGen}, \mathcal{P}, \mathcal{V}$, and we omit them from the descriptions.

$(\mathsf{pk}, \mathsf{help}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$**:** Outputs the public key pk, a helper help and a secret key sk. The value help is analyzed as a public key with respect to security (i.e. it need not be kept secret and is given to the adversary) but is thought of as a secret key for *usability* (i.e. it is not used by honest verifiers).[1]

$\mathcal{P}(\mathsf{pk}, \mathsf{help}, \mathsf{sk}), \mathcal{V}(\mathsf{pk})$**:** These are the prover and verifier ITMs respectively. The verifier $\mathcal{V}$ outputs a judgement from one of $\{Accept, Reject\}$ at the conclusion of a protocol execution.

We require that an ID scheme is *complete*, so that in an interaction $\{\mathcal{P}(\mathsf{pk}, \mathsf{sk}) \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ between honest prover and honest verifier, the verifier *always accepts* the proof. We now formally define what it means for an ID scheme to be leakage resilient. As discussed, we will consider two separate security notions. The first notion, called *pre-impersonation*

[1] In some of our constructions, when sk is made intentionally huge, the size of help will become large as well, and thus it is important that this does *not* detract from the usability of the scheme by also increasing the size of the public key.

leakage security, is modeled by the attack game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$ and only allows the adversary to submit leakage queries *prior to* an impersonation attack, but not during one. The second notion, called *anytime leakage security*, is modeled by the attack game $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$ where the adversary can perform leakage attacks adaptively at any point in time, even during an impersonation attack. The two attack games are defined below and only differ in the impersonation stage.

$\underline{\mathsf{IDPRE}_\ell^\lambda(\mathcal{A}),\ \ \mathsf{IDANY}_\ell^\lambda(\mathcal{A})}$

1. Key Stage: Let $\mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{help}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$ and give $(\mathsf{params}, \mathsf{pk}, \mathsf{help})$ to $\mathcal{A}$.

2. Test Stage: The adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot),\mathcal{P}(\mathsf{pk},\mathsf{sk})}$ gets access to the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and to an honest prover $\mathcal{P}(\mathsf{pk}, \mathsf{sk})$, modeled as an oracle that runs (arbitrarily many) proofs upon request.

3. Impersonation Stage: This stage is defined separately for the two games.

For $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$: The adversary $\mathcal{A}$ *loses* access to the all oracles and runs a protocol $\{\mathcal{A} \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ with as an honest verifier.

For $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$: The adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)}$ *maintains* access to the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$, but *not* the prover oracle $\mathcal{P}$, and runs a protocol $\{\mathcal{A}^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)} \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ with an honest verifier.

The *advantage* of an adversary $\mathcal{A}$ in the games $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$, $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$ is the probability that the verifier $\mathcal{V}$ accepts in the impersonation stage.

Definition 4. *Let* $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *be an identification scheme with perfect completeness, parameterized by security parameter* λ*. We say that the scheme is* secure with pre-impersonation leakage ℓ *if the advantage of any PPT adversary* $\mathcal{A}$ *in the attack game* $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$ *is negligible in* λ*. We say that the scheme is* secure with anytime leakage ℓ *if the above also holds for the attack game* $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$.

4.2 Construction 1: Generalized Okamoto Scheme

We now show that the Okamoto identification scheme from [25] is secure against key leakage attacks. The standard Okamoto scheme is defined with respect to two generators. Here, we describe a generalized version of the Okamoto scheme with m generators. Since we will re-use the basic components of the scheme as building-blocks for our more complicated schemes, we abstract away most of the computation of the scheme into the algorithms $(\mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ which are used by $\mathcal{P}, \mathcal{V}$ to run the protocol as defined in Figure 1. To analyze the above scheme, we define the relation $\mathcal{R} = \{(\mathsf{pk}, \mathsf{sk}) | \mathsf{sk} = (x_1, \ldots, x_m), \mathsf{pk} = \prod_{j=1}^m g_j^{x_j}\}$. We will rely on only three properties of the relation $\mathcal{R}$ and the generalized Okamoto ID scheme, outlined in Lemma 3.

Lemma 3. *The following three properties hold for the* $\mathrm{Okamoto}_m^\lambda$ *ID scheme:*

(1) The protocol $(\mathcal{P}, \mathcal{V})$ *is a* Σ*-protocol for the relation* $\mathcal{R}$*.*

(2) Denoting key pairs as random variables, we get $\widetilde{\mathbf{H}}_\infty(\mathsf{SK}|\mathsf{PK}) \geq (m-1)\log(p)$.

```
ParamGen$(1^\lambda)$ : Let $(p, G, g) \leftarrow \mathcal{G}(1^\lambda), g_1, \ldots, g_m \leftarrow_R G$.
    Set params $= (p, G, g_1, \ldots, g_m)$.
KeyGen(): Let sk $= (x_1, \ldots, x_m) \leftarrow_R (\mathbb{Z}_p)^m$, pk $= \prod_{j=1}^m \{g_j\}^{x_j}$. Output (pk, $\perp$, sk).
$\mathcal{P}, \mathcal{V}$: The machines $\mathcal{P}, \mathcal{V}$ run the following protocol:
    (1) $\mathcal{P}$: Computes $(a, \overline{y}) \leftarrow \mathbf{A}()$ and sends $a$ to $\mathcal{V}$.
        $\mathbf{A}()$ : Let $\overline{y} = (y_1, \ldots, y_m) \leftarrow_R (\mathbb{Z}_p)^m$, $a = \prod_{j=1}^m {g_j}^{y_j}$. Output $(a, \overline{y})$.
    (2) $\mathcal{V}$: Choose $c \leftarrow_R \mathbb{Z}_p$ and send $c$ to $\mathcal{P}$.
    (3) $\mathcal{P}$: Compute $\overline{z} \leftarrow \mathbf{Z}_{\mathsf{sk}}(c, \overline{y})$ and send $\overline{z}$ to $\mathcal{V}$.
        $\mathbf{Z}_{\mathsf{sk}}(c, \overline{y})$: Compute $z_j := y_j + cx_j$ for $j = 1, \ldots, m$, output $\overline{z} := (z_1, \ldots, z_m)$.
    $\mathbf{Ver}_{\mathsf{pk}}(a, c, \overline{z})$: Output Accept iff $\prod_{j=1}^m g_j^{z_j} \overset{?}{=} a(\mathsf{pk})^c$.
```

Fig. 1. The Okamoto_m^λ identification scheme

(3) *Under the discrete logarithm assumption, it is difficult to find a public key* pk *and two* different *secret keys* sk′ ≠ sk *for* pk. *In particular, for any PPT adversary $\mathcal{A}$:*

$$\Pr\left[\mathsf{sk}' \neq \mathsf{sk} \textit{ and } (\mathsf{pk}, \mathsf{sk}'), (\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}, \mathsf{sk}') \leftarrow \mathcal{A}(\mathsf{params}) \\ \mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda) \leq \mathrm{negl}(\lambda)\end{array}\right].$$

Using the properties in the above lemma, we show that the Okamoto ID scheme is secure against key leakage attacks.

Theorem 1. *Under the* DL *assumption,* Okamoto_m^λ *is a secure ID-scheme for pre-impersonation leakage of up to* $\ell = (m-1)\log(p) - \lambda \geq (1 - \frac{2}{m})|\mathsf{sk}|$ *bits. It is secure with anytime leakage of up to* $\ell' = \frac{1}{2}\ell$ *bits.*

Proof Sketch. Assume that there is an adversary $\mathcal{A}$ that has a non-negligible advantage in the pre-impersonation leakage attack game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$. Then there is a reduction which, for randomly chosen params, finds two distinct secret keys sk, sk′ for a single public-key pk (contradicting part (3) of Lemma 3). In particular, the reduction chooses a random (pk, sk) tuple and uses sk to simulate the leakage oracle, and the honest-prover oracle for the attacker $\mathcal{A}$ during the "test stage". Then, during the impersonation stage, the reduction runs $\mathcal{A}$ twice, with two randomly chosen challenges c, c' (using rewinding). There is a non-negligible probability that $\mathcal{A}$ produces two accepting conversations $(a, c, z), (a, c', z')$ with $c \neq c'$. Using the special soundness property of the Σ-protocol, the reduction uses these two conversation to recover a secret key sk′.

We must now analyze the probability of sk = sk′. We think of the reduction as an experiment $\mathcal{E}_0$ where $\mathcal{A}$ gets PK and access to the oracles $\mathcal{O}_{\mathsf{SK}}^{\lambda,\ell}(\cdot), \mathcal{P}(\mathsf{PK}, \mathsf{SK})$. Let $\mathcal{E}_1$ be the same experiment as $\mathcal{E}_0$, except that the predictor does not get access to $\mathcal{O}_{\mathsf{SK}}^{\lambda,\ell}(\cdot)$, and $\mathcal{E}_2$ be the same as $\mathcal{E}_1$ except that the predictor doesn't get access to $\mathcal{P}(\mathsf{PK}, \mathsf{SK})$ either (i.e. only gets PK). Then

$$\begin{aligned}\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_0) &\geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) - \ell \\ &\geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_2) - \ell = \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathsf{PK}) - \ell \\ &\geq (m-1)\log(p) - \ell \geq \lambda\end{aligned}$$

where the first inequality follows by Lemma 2, the second one by Lemma 1, and the last one by part (3) of Lemma 3. The probability of the reduction outputting $\mathsf{sk}' = \mathsf{sk}$ is therefore upper bounded by $2^{-\lambda}$ and hence, with non-negligible probability, the reduction produces two distinct secret keys $\mathsf{sk} \neq \mathsf{sk}'$.

For anytime leakage, $\mathcal{A}$ can make calls to the leakage oracle for ℓ' bits *even* during the impersonation stage. Since the reduction runs the impersonation stage twice on different challenges (by rewinding $\mathcal{A}$), the reduction needs $2\ell'$ bits of leakage. Therefore, we can only handle $\ell' = \frac{1}{2}\ell$ bits of anytime leakage. □

4.3 Construction 2: Adding Flexibility through Direct-Products

We now propose a construction of a leakage-resilient ID scheme with pre-impersonation security, that is suitable for the BRM setting. In particular, it is possible to increase the allowed leakage ℓ arbitrarily without significantly affecting the communication and computation complexity, or even the size of the public key and public parameters. As we will see, some of the parameters in our construction are still sub-optimal, and we will get further efficiency gains in Section 4.4. However, the construction we present here is more natural and simpler to understand, and hence we present it first.

The main idea of our construction, is to run many copies of the Okamoto_m^λ scheme in parallel. In particular, the secret key will be a database $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$ where each $\mathsf{sk}[i]$ is a secret key for the underlying generalized Okamoto scheme, and defines a corresponding public key $\mathsf{pk}[i]$. During key generation, the prover also chooses a key pair $(\mathsf{verk}, \mathsf{sigk})$ for a signature scheme and computes signatures $\sigma[i]$ for each public key $\mathsf{pk}[i]$ and sets the helper string to $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$ (after which point sigk is never used again and deleted from memory). We could then define a four-round protocol, where the verifier begins by giving t random indices $(r_1, \ldots, r_t) \in [n]^t$ to the prover. Then the prover and verifier then execute t independent copies of Okamoto_m^λ (in parallel) for the public keys $\mathsf{pk}[r_1], \ldots, \mathsf{pk}[r_t]$, which the prover sends to the verifier along with their signatures $\sigma[r_i]$. Our actual construction is a three-round scheme where the indices $r_1, \ldots, r_t$ are sent by the verifier with the challenge and we rely on the fact that the first messages a of the generalized Okamoto scheme does not depend on the public key pk.

To analyze the security of the scheme, we notice that, in a pre-impersonation attack, the adversary's queries to the leakage oracle must be independent of the indices $r_1, \ldots, r_t$. In the full version, we show that, if sk has a significant amount of entropy at the beginning of the impersonation stage, then the random tuple $(\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t])$ will *preserve* some significant amount of this entropy as well. This analysis is based on thinking of the tuples $(\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t])$ as positions in an (exponentially) long *direct-product encoding* of sk. Such codes were defined and analyzed in [16], where it is shown that they are "approximately-list decodable". We show that this property implies entropy preservation in our sense. Our security analysis then relies on the fact that, if an adversarial prover can complete a proof on the challenge $r_1, \ldots, r_t$, then it must *know* the values $(\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t])$ in their entirety, which is unlikely by our entropy argument.

We note that, although our discussion seems quite general, it is not clear that the main idea of our construction (taking direct products) would imply a general a compiler

```
ParamGen($1^\lambda$): Let $(p, G, g) \leftarrow \mathcal{G}(1^\lambda), g_1, \ldots, g_m \leftarrow_R G$.
    Set params $= (p, G, g_1, \ldots, g_m)$.
KeyGen(): Choose $(\mathsf{verk}, \mathsf{sigk}) \leftarrow \mathsf{SigKeyGen}(1^\lambda)$ and set $\mathsf{pk} = \mathsf{verk}$.
    For $i = 1, \ldots, n$ set: $(\mathsf{sk}[i], \mathsf{pk}[i]) \leftarrow \mathbf{Gen}(), \sigma[i] = \mathsf{Sign}_{\mathsf{sigk}}(i||\mathsf{pk}[i])$.
    Set $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$, $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$.
    Output $(\mathsf{pk}, \mathsf{sk}, \mathsf{help})$.†
$\mathcal{P}, \mathcal{V}$: The machines $\mathcal{P}, \mathcal{V}$ run the following protocol:
    (1) $\mathcal{P}$: For $i = 1, \ldots, t$: choose $(a_i, \overline{y}_i) \leftarrow \mathbf{A}()$. Send $(a_1, \ldots, a_t)$ to $\mathcal{V}$.
    (2) $\mathcal{V}$: Choose $t$ indices $(r_1, \ldots, r_t) \leftarrow_R [n]^t$, and $c^* \leftarrow_R \mathbb{Z}_p$.
        Send the challenge $c = (r_1, \ldots, r_t, c^*)$ to $\mathcal{P}$.
    (3) $\mathcal{P}$: For $i = 1, \ldots, t$: set $\mathsf{pk}_i = \mathsf{pk}[r_i]$, $\sigma_i = \sigma[r_i]$, $\overline{z}_i = \mathbf{Z}_{\mathsf{sk}[r_i]}(c^*, \overline{y}_i)$ and send
        $(\mathsf{pk}_i, \sigma_i, \overline{z}_i)$ to $\mathcal{V}$.
    $\mathcal{V}$ accepts iff, for $i = 1, \ldots, t$:
    (I) The conversation $(a_i, c^*, \overline{z}_i)$ is accepting for $\mathsf{pk}_i$. That is, $\mathbf{Ver}_{\mathsf{pk}_i}(a_i, c^*, \overline{z}_i) \stackrel{?}{=}$ Accept.
    (II) The signatures $\sigma_i$ for $r_i||\mathsf{pk}_i$ verify under $\mathsf{pk}$. That is $\mathsf{SigVer}_{\mathsf{pk}}(r_i||\mathsf{pk}_i, \sigma_i) \stackrel{?}{=}$ Accept.
```

† Note that the values $\mathsf{pk}[i]$ can be easily computed from $\mathsf{sk}[i]$ and thus need not be stored separately.

Fig. 2. The $\mathrm{DirProd}^{\lambda}_{n,m,t}$ identification scheme

which converts an ID scheme with pre-impersonation leakage ℓ into one with "amplified" pre-impersonation leakage $\ell' \gg \ell$. Indeed, our argument is (crucially) information theoretic in the sense that we show that a random subset of secret keys still has (information theoretic) entropy after the adversary gets some key leakage. To translate this into a more general argument, we would need to somehow simulate an ℓ' bit leakage oracle for the entire key sk by accessing (many) ℓ-bit leakage oracles for the individual keys $\mathsf{sk}[i]$, which does not seem possible.

We present our construction, called $\mathrm{DirProd}^{\lambda}_{n,m,t}$ in Figure 2. The presentation is based on the algorithms $(\mathbf{Gen}, \mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ where $\mathbf{Gen}$ is the key generation algorithm for the underlying Okamoto scheme, and $(\mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ are the algorithms used by the prover and verifier as defined in Figure 1.

Theorem 2. *Assuming that* $(\mathsf{SigKeyGen}, \mathsf{Sign}, \mathsf{SigVer})$ *is an existentially secure signature scheme under chosen message attacks, and assuming the hardness of* DL*, the construction* $\mathrm{DirProd}^{\lambda}_{n,m,t}$ *is a secure ID-scheme for pre-impersonation leakage of up to* $\ell = (1-\delta)nm\log(p) = (1-\delta)|\mathsf{sk}|$ *bits where* $\delta = \frac{1}{m}(1 + \frac{\log(n)}{\lambda} + \frac{4}{n}) + \frac{2\lambda}{t}$ *which approaches* $\frac{1}{m} + O(\lambda/t)$*.*

4.4 Construction 3: Saving Communication Using Compressed Direct-Products

As we saw, Construction 2 gives us flexibility, in the sense that we can increase the parameter n to allow for arbitrarily large leakage ℓ, without (significantly) affecting the size of the public key, the computation or the communication complexity. Unfortunately, even though these factors do not depend on n, the communication of the scheme

$\mathsf{ParamGen}(1^\lambda)$**:** Choose $(p, G, g) \leftarrow \mathcal{G}(1^\lambda)$, $(g_1, \dots, g_m, u) \leftarrow_R G^{m+1}$.
Set $\mathsf{params} = (p, G, g_1, \dots, g_m, u)$.
$\mathsf{KeyGen}()$**:** Choose $s \leftarrow_R \mathbb{Z}_p$ and set $\mathsf{pk} = v = u^s$.
Choose $(\mathsf{pk}[i], \mathsf{sk}[i]) \leftarrow_R \mathbf{Gen}()$ and set $\sigma[i] = (H(i)\mathsf{pk}[i])^s$ for $i \in \{1, \dots, n\}$.[†]
Set $\mathsf{sk} = (\mathsf{sk}[1], \dots, \mathsf{sk}[n])$, $\mathsf{help} = (\sigma[1], \dots, \sigma[n])$. Output $(\mathsf{pk}, \mathsf{sk}, \mathsf{help})$.
$\mathcal{P}, \mathcal{V}$**:** The machines $\mathcal{P}, \mathcal{V}$ run the following protocol:
(1) $\mathcal{P}$**:** Choose $(a, \overline{y}) \leftarrow \mathbf{A}()$ and send a to $\mathcal{V}$.
(2) $\mathcal{V}$**:** Choose t indices $(r_1, \dots, r_t) \leftarrow_R [n]^t$, and $(c^*, e) \leftarrow_R (\mathbb{Z}_p)^2$.
Send the challenge $c = (r_1, \dots, r_t, e, c^*)$ to $\mathcal{P}$. [‡]
(3) $\mathcal{P}$**:** Compute $\mathsf{sk}^* = (x_1^*, \dots, x_m^*)$ where $\{x_j^* = \sum_{i=1}^t (x_j[r_i]) e^{i-1}\}_{j \in \{1,\dots,t\}}$.
Set $\mathsf{pk}^* = \prod_{i=1}^t \mathsf{pk}[r_i]^{(e^{i-1})}$, $\sigma^* = \prod_{i=1}^t \sigma[r_i]^{(e^{i-1})}$, $\overline{z} = \mathbf{Z}_{\mathsf{sk}^*}(c^*, \overline{y})$.
Send $(\mathsf{pk}^*, \sigma^*, \overline{z})$ to $\mathcal{V}$.
$\mathcal{V}$ accepts iff:
(I) The conversation $(a, c^*, \overline{z})$ is accepting for pk^*. That is, $\mathbf{Ver}_{\mathsf{pk}^*}(a, c^*, \overline{z}) = \mathit{Accept}$.
(II) The value $(u, v, (\mathsf{pk}^* \prod_{i=1}^t H(r_i)^{e^{i-1}}), \sigma^*)$ is a DDH tuple.

† Recall that we write $\mathsf{sk}[i] = (x_1[i], \dots, x_m[i]) \in \mathbb{Z}_p^m$.
‡As stated, the challenge size is $t\log(n)$ which dominates the remaining communication. In the Random Oracle model, we can compress the challenge to a λ bit value, which is then expanded into the full challenge using the Random Oracle. This version matches the parameters claimed in Table 1.

Fig. 3. The $\mathrm{CompDirProd}_{n,m,t}^{\lambda}$ identification scheme

is fairly large since it uses $t = O(\lambda)$ copies of the underlying Okamoto scheme. In fact, just having the prover send t public keys $\mathsf{pk}[r_1], \dots, \mathsf{pk}[r_t]$ to the verifier in construction 2 already takes the communication complexity to $O(\lambda^2)$, which may be prohibitive. As we will see later, large communication complexity of the ID schemes will translate into long Fiat-Shamir signatures and, therefore, large communication complexity in our final authenticated key agreement protocols.

In this section, we show how to reduce the communication complexity of the ID scheme significantly. As in Construction 2, the secret key $\mathsf{sk} = (\mathsf{sk}[1], \dots, \mathsf{sk}[n])$ is a (possibly huge) database of keys $\mathsf{sk}[i]$ for the underlying generalized Okamoto scheme, and the verifier selects a random set of t indices which define a set of t secret keys $\mathsf{sk}[r_1], \dots, \mathsf{sk}[r_t]$ used by the protocol execution. However, instead of running parallel versions of the $\mathrm{Okamoto}_m^\lambda$ scheme for these keys individually, the prover now *compresses* them into a *single secret key* sk^* and then runs a *single copy* of the Okamoto scheme for the corresponding public key pk^*, which the prover sends to the verifier. The two important properties of this compression are: (1) it must be entropy preserving, in the sense that sk^* should be (information theoretically) unpredictable, assuming that there is sufficient entropy spread-out over the entire database sk and (2) the public key pk^* for the secret key sk^* can be computed from $\mathsf{pk}[r_1], \dots, \mathsf{pk}[r_t]$ alone, so that the values pk^* do not decrease the entropy of the database sk.

Our compression function is based on the *Reed-Solomon Error-Correcting Code*. In particular, the verifier chooses a random value $e \in \mathbb{Z}_p$, and the prover compresses the t secret keys $\{\mathsf{sk}[r_i] = (x_1[r_i], \dots, x_m[r_i])\}_{i \in \{1,\dots,t\}}$ into a single key $\mathsf{sk}^* =$

$(x_1^*, \ldots, x_m^*)$, where $x_j^* = \sum_{i=1}^{t}(x_j[r_i])e^{(i-1)}$ is the e-th position in the Reed-Solomon encoding of the value $(x_j[r_1], \ldots, x_j[r_t])$. In the full version of this paper, we show that this compression function is entropy preserving. The corresponding public key pk^* for sk^* is just $\mathsf{pk}^* = \prod_{i=1}^{t}(\mathsf{pk}[r_i])^{\left(e^{i-1}\right)}$, which is easy to compute from the individual keys $\mathsf{pk}[r_i]$. Thus it satisfies the two properties we required.

Of course, there is one crucial problem we have not yet addressed: how does an honest verifier check that the compressed public key pk^* given by the prover is indeed the right one (i.e. corresponds to the correct combination of $\mathsf{pk}[r_1], \ldots, \mathsf{pk}[r_t]$ using e as requested)? We can no longer use signatures, as in construction 2, since the number of possibilities for pk^* is exponential. Instead, we use a modification of the BLS signature scheme ([4]) to compute "helper values" $\sigma[i]$, which can be efficiently combined into a short "authenticator" σ^*. The authenticator σ^* essentially ensures that the adversary sends the correct public key pk^*. We present our construction, in Figure 3. The presentation is based on the algorithms $(\mathbf{Gen}, \mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ for the underlying generalized Okamoto scheme (see Figure 1). In addition, our construction relies on a hash function H modeled as a random oracle. The security of the scheme is formalized in Theorem 3. The proof appears in the full version of this paper, and requires a careful analysis, combining the authentication properties of the modified BLS signatures with the rewinding strategy for the Okamoto scheme.

Theorem 3. *Under the* GDH *assumption, the* $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$ *scheme is a secure ID-scheme in the Random Oracle model, with pre-impersonation leakage of up to* $\ell = (1-\delta)nm\log(p) = (1-\delta)|\mathsf{sk}|$ *bits where* $\delta = \frac{1}{m}(1 + \frac{\log(n)}{\lambda} + \frac{10}{n}) + \frac{6\lambda}{t}$ *which approaches* $\frac{1}{m} + O(\lambda/t)$.

5 Existentially and Entropically Unforgeable Signatures

We now look at leakage-resilient signatures, where the adversary can (periodically) query a leakage oracle for up to ℓ bits of information about the secret key. Unfortunately, if ℓ is larger than the size of a single signature, it is clear that we cannot achieve the standard notion of *existential unforgeability* as the attacker can simply choose to learn the a signature of some message m as its leakage function. Therefore, to construct meaningful signature schemes in the BRM, we also define a new (weaker) security notion called *entropic unforgeability*, where an attacker should be unable to forge messages which are chosen from some distribution of significant entropy and given to the adversary only *after* the leakage attack. To further strengthen the attack game we let the forger select this distribution. This notion is useful since, in many practical scenarios, an attacker must be able to forge signatures for messages that are somehow beyond her control, in order to damage the security of the system.

A signature scheme consists of four algorithms: $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. To capture entropic unforgeability, we separate the attacker into two parts $\mathcal{A}=(\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ runs during the first stage of the attack, with access to a leakage oracle and signing oracle. Once this stage is done, $\mathcal{A}_1$ can output an arbitrary hint for $\mathcal{A}_2$, who then attempts to forge the signature of some message while having access *only* to the signing oracle. The formal definition of the *unforgeability attack game* $\mathsf{EUG}^{\lambda}_{\ell}$ appears

$\underline{\mathsf{EUG}_\ell^\lambda}$

Initialization: The challenger selects $(\mathsf{verk}, \mathsf{help}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives verk to the forger $\mathcal{A}_1$.

Signing & Leakage Queries: Adversary $\mathcal{A}_1^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot), \mathcal{S}_{\mathsf{sigk}}(\cdot)}$ is given access to the signing oracle $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ and leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and outputs an arbitrary hint $v \in \{0,1\}^*$.

Post-Leakage: Adversary $\mathcal{A}_2^{\mathcal{S}_{\mathsf{sigk}}(\cdot)}$ is given the hint v and access to (only) the signing oracle $\mathcal{S}_{\mathsf{sigk}}(\cdot)$. We parse the output of $\mathcal{A}_2$ as a message, signature pair (m, σ).

Fig. 4. Entropic/Existential Unforgeability Attack Game

in Figure 4. We use $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ to denote the *signing oracle*, which, on input $m \in \{0,1\}^*$, outputs $\sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(m)$. We define the advantage of forger $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ to be the probability that $\mathsf{Verify}_{\mathsf{verk}}(\mathsf{m}, \sigma) = Accept$ and that the signing oracle was never queried with m. For entropic security, we also require that the output message m is chosen sufficiently randomly by $\mathcal{A}_2$, so that it could not have been predicted by $\mathcal{A}_1$.

Definition 5. *For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, let $\mathsf{View}_{\mathcal{A}_1}$ be a random variable describing the view of $\mathcal{A}_1$ including its random coins and signing-oracle/leakage-oracle responses.*[2] *Let $\mathsf{MSG}_{\mathcal{A}_2}$ be the random variable describing the message output by $\mathcal{A}_2$ in $\mathsf{EUG}_\ell^\lambda$. We say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is* entropic *if $\widetilde{\mathbf{H}}_\infty(\mathsf{MSG}_{\mathcal{A}_2} | \mathsf{View}_{\mathcal{A}_1}) \geq \lambda$ for security parameter λ. We say that a signature scheme $(\mathsf{KeyGen}, \mathsf{Verify}, \mathsf{Sign})$ is* existentially unforgeable *with leakage ℓ if the advantage of any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the game $\mathsf{EUG}_\ell^\lambda(\mathcal{A})$ is negligible in λ. We say that the signature scheme is* entropically unforgeable *with leakage ℓ if the above only holds for* entropic *adversaries.*

We use the Fiat-Shamir heuristic [14] to construct entropically (resp. existentially) unforgeable signature schemes secure against ℓ bits of key leakage, from ID schemes secure against pre-impersonation (resp. anytime) leakage of ℓ bits. Recall that, for a three round ID scheme with flows (a, c, z), the Fiat-Shamir signature scheme defines a signature of a message m to be (a, z) such that the conversation $(a, H(a||m), z)$ is accepting. Here $H(\cdot)$ is hash function modeled as a Random Oracle.

Theorem 4. *Let ID by a public coins ID scheme consisting of three rounds of interaction and let Sig be the signature scheme produced by the Fiat-Shamir heuristic applied to ID. (1) If ID allows $\underline{\text{pre-impersonation}}$ leakage ℓ, then Sig is $\underline{\text{entropically}}$ unforgeable with leakage ℓ. (2) If ID allows $\underline{\text{anytime}}$ leakage ℓ, then Sig is $\underline{\text{existentially}}$ unforgeable with leakage ℓ.*

CONCRETE SCHEMES: Combining this theorem with the $\mathrm{Okamoto}_m^\lambda$ identification scheme, analyzed in Theorem 1, we obtain a (I) leakage-resilient existentially unforgeable signature scheme where ℓ approaches up to half the size of the secret key (and signature) and a (II) leakage-resilient entropically unforgeable signature scheme where

[2] In the Random Oracle Model, this also includes responses to Random Oracle queries.

ℓ approaches the size of the entire secret key (and signature). For the BRM setting, we can instead use $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$, analyzed in Theorem 3, and get a (III) entropically unforgeable signatures, where ℓ approaches the size of the entire secret key, and can be made arbitrarily large without negatively impacting the other parameters (and can be much larger than the size of a signature).

6 Interactive Encryption, Authentication and AKA

Using leakage-resilient entropically unforgeable signatures, we can construct several *interactive* leakage-resilient primitives including encryption, authentication and authenticated key agreement. The security of these interactive primitives is preserved even if the adversary gets up to ℓ bits of leakage *prior to* the start of protocol execution, and the key is leaked entirely *after* the end of protocol execution.

For example consider the following simple two-round interactive authentication protocol. The verifier sends a random challenge r to the signer, who returns the signature $\sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(m||r)$ for a message m. If no leakage occurs between the time where the verifier sends r and receives σ then, since r is random, entropic-unforgeability ensures that the adversary cannot forge the signature σ' of $m'||r$ for some $m' \neq m$.

Alternatively, consider the following simple three-round interactive encryption. The sender sends a random challenge r to the receiver, who in turn chooses a fresh temporary public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ for a standard (non-leakage resilient) encryption scheme and sends $\mathsf{pk}, \sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(\mathsf{pk}||r)$ to the sender. If the signature verifies, the sender sends an encryption of the message m under pk to the receiver, who decrypts it with sk and deletes all temporary state (including sk) immediately afterwards. This way, if no leakage occurs between the time that r is sent and sk is deleted, then entropic-unforgeability ensures that sk was chosen by the honest receiver, and so privacy is preserved since the adversary can never learn anything about sk.

Defining the security of interactive encryption and authentication schemes is a tedious process. Therefore, in the full version of the paper, we concentrate on the single primitive of Authenticated Key-Agreement (AKA), which allows for interactive encryption as well as authentication. We adapt the notion of *SK-security with perfect forward secrecy* from Canetti and Krawczyk in [6], and update it to model key-leakage attacks. We then analyze a simple AKA construction from [6], which essentially consists of the Diffie-Hellman key-exchange protocol in which the parties sign the exchanged messages together with unique session information so as to bind the protocol execution to a particular session. We show that, if we employ leakage-resilient entropically secure signatures in this construction, then the resulting AKA is leakage-resilient as well.

7 Invisible Key Updates

Our schemes allow for efficient updates of the secret key, using an externally stored "master update key", so that the adversary is only limited to leaking ℓ bits between updates, but can get unlimited leakage overall. Since this is technically simple, we only give a high-level description of how this is done.

In particular, for our constructions $\mathrm{DirProd}^{\lambda}_{n,m,t}$ and $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$, there is already a "master key" which is used to create a secret-key database of unbounded size – namely, the "master signing key" for generic signatures in $\mathrm{DirProd}^{\lambda}_{n,m,t}$ and for modified BLS signatures in $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$. In our original descriptions, this master key is used once to create the secret-key database $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$ and the helper $\mathsf{help} = (\mathsf{help}[1], \ldots, \mathsf{help}[n])$, and is then deleted immediately afterwards. However, we notice that the master key can really generate arbitrarily many secret keys $(\mathsf{sk}[1], \mathsf{sk}[2], \ldots)$ and corresponding helper strings $(\mathsf{help}[1], \mathsf{help}[2], \ldots)$.

To perform updates, we can store this "mater update key" on a separate external device, which is not susceptible to leakage, as a "mater update key". To update the secret-key database, we simply overwrite the current secret-keys and helper values with the "next" n values so that $\mathsf{sk} := (\mathsf{sk}[nk+1], \ldots, \mathsf{sk}[n(k+1)]), \mathsf{help} = (\mathsf{help}[nk+1], \ldots, \mathsf{help}[n(k+1)])$ after the kth update. To run an ID scheme, the prover simply sends the current index k to the verifier in the first flow of the protocol, and the verifier chooses the challenge indices in the range $[nk+1, n(k+1)]$. Note that an adversarial prover can send any index k' of his choosing. However, if the adversary learns at most ℓ bits in between any two updates, then there is *no* index k' for which the adversary can successfully run an impersonation attack.

The above updates for ID schemes translate to similar updates for our signature schemes and AKA protocols. Notice that these updates do not modify the public key, and the user has a completely free choice of when or how often the secret key is updated. However, it is important that the "master signing key" is stored securely and that the adversary cannot get any leakage of this key.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC, pp. 474–495 (2009)
2. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: ACM Conference on Computer and Communications Security, pp. 244–250 (1993)
3. Bennett, C.H., Brassard, G., Robert, J.-M.: Privacy amplification by public discussion. SIAM J. Comput. 17(2), 210–229 (1988)
4. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
5. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
6. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
7. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)
8. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)

9. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC (to appear, 2009)
10. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
11. Dodis, Y., Sahai, A., Smith, A.: On perfect and adaptive security in exposure-resilient cryptography. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 301–324. Springer, Heidelberg (2001)
12. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
13. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302 (2008)
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
15. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium, pp. 45–60 (2008)
16. Impagliazzo, R., Jaiswal, R., Kabanets, V.: Approximately list-decoding direct product codes and uniform hardness amplification. In: FOCS, pp. 187–196 (2006)
17. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
18. Kalai, Y.T., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs and applications. Personal Communication (2009)
19. Katz, J.: Signature schemes with bounded leakage resilience. Cryptology ePrint Archive, Report 2009/220 (2009), `http://eprint.iacr.org/2009/220`
20. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
21. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
22. Maurer, U.M.: Protocols for secret key agreement by public discussion based on common information. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 461–470. Springer, Heidelberg (1993)
23. Micali, S., Reyzin, L.: Physically observable cryptography. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
24. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (to appear, 2009), `http://eprint.iacr.org/2009/105`
25. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
26. Ong, H., Schnorr, C.-P.: Fast signature generation with a fiat shamir-like scheme. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 432–440. Springer, Heidelberg (1991)
27. Pietrzak, K.: A leakage-resilient mode of operation. In: Eurocrypt 2009, Cologne, Germany (2009)
28. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (ema): Measures and countermeasures for smart cards. In: E-smart, pp. 200–210 (2001)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)

Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate

Marc Stevens[1], Alexander Sotirov[2],
Jacob Appelbaum[3], Arjen Lenstra[4,5], David Molnar[6],
Dag Arne Osvik[4], and Benne de Weger[7]

[1] CWI, Amsterdam, The Netherlands
[2] `http://www.phreedom.org`
[3] `http://www.appelbaum.net`
[4] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
[5] Alcatel-Lucent Bell Laboratories
[6] University of California at Berkeley
[7] EiPSI, TU Eindhoven, The Netherlands
[1–7] `md5-collisions@phreedom.org`

Abstract. We present a refined chosen-prefix collision construction for MD5 that allowed creation of a rogue Certification Authority (CA) certificate, based on a collision with a regular end-user website certificate provided by a commercial CA. Compared to the previous construction from Eurocrypt 2007, this paper describes a more flexible family of differential paths and a new variable birthdaying search space. Combined with a time-memory trade-off, these improvements lead to just three pairs of near-collision blocks to generate the collision, enabling construction of RSA moduli that are sufficiently short to be accepted by current CAs. The entire construction is fast enough to allow for adequate prediction of certificate serial number and validity period: it can be made to require about 2^{49} MD5 compression function calls. Finally, we improve the complexity of identical-prefix collisions for MD5 to about 2^{16} MD5 compression function calls and use it to derive a practical single-block chosen-prefix collision construction of which an example is given.

Keywords: MD5, collision attack, certificate, PlayStation 3.

1 Introduction

At Eurocrypt 2007, it was shown how chosen-prefix collisions for MD5 can be constructed and an undesirable consequence for any public key infrastructure (PKI) was pointed out in the form of different certificates with the same valid signature (cf. [13]). Actual realization of the threat in question was considered to be hard due to a combination of difficulties, some related to the construction, others to the way certificates are produced by CAs. Thus, some CAs kept using MD5, either consciously based on the perception that the obstacles were too high, or because they were unaware of lurking dangers.

It was found, however, that for at least one commercial CA the relevant obstacles could be overcome with non-negligible probability. Understandably, this triggered new research in the earlier chosen-prefix collision construction. A couple of non-trivial refinements removed all remaining obstacles, thereby in principle allowing us to create real havoc.

Obviously, creating havoc was not our goal. It was our intention and priority that all relevant responsible parties would develop a thorough understanding of the implications of chosen-prefix collisions for MD5. Furthermore, before publishing the details of our results, we wanted to make sure that all parties would have had both a strong impetus and ample time to adequately change their procedures. Therefore, we decided to actually implement our construction and to try and exploit it in practice by attempting to create a harmless rogue CA certificate that would be accepted by all regular web browsers: harmless, because they would only do so after setting their date back to August 2004, because we would keep the private key of the rogue CA in question under tight control, and because we would not right away reveal the details of our method. After a moderate number of attempts we succeeded to create such a certificate.

The announcement of our successful creation of a rogue CA certificate had the desired effect. CAs and other vendors responded swiftly and adequately. We believe that as a result of our exercise, the bar to undermine the security of PKI was raised, somewhat. Given that the current situation with respect to usage of MD5 looks much better than when we made our announcement, we feel that the details behind our method can now be revealed. We also feel that this should indeed be done to give others the opportunity to further build on them and to develop a better understanding of the lack of strength of currently popular cryptographic hash functions. Fully appreciating the details presented here requires a full understanding of the approach from [13].

We describe, roughly, what was achieved in the Eurocrypt 2007 paper [13] and why those methods were believed to have limited impact. Given any two chosen message prefixes P and P', it was shown how suffixes S and S' can be constructed such that the concatenations $P\|S$ and $P'\|S'$ collide under MD5. In the X.509 certificate context, the prefixes include the Distinguished Name fields, and the suffixes are the initial parts of the RSA moduli. A simple, previously published method was then used to construct a further extension T such that each of $P\|S\|T$ and $P'\|S'\|T$ is a complete to-be-signed part, with two different hard to factor RSA moduli contained in $S\|T$ and $S'\|T$, respectively. Because the two to-be-signed parts still collide under MD5, this allowed construction of two X.509 certificates with identical MD5-based signatures but different Distinguished Names and different public keys. Put differently, assuming full control over the prefix part P and RSA public key data of a legitimate user, a certificate of that user's data can be used to fraudulently obtain a rogue certificate for any party identified by a prefix part P' selected by the attacker. Using moderate resources, the calculation of suffixes S, S' and T, given any chosen prefixes P and P', can be completed in a day using e.g. a quad-core PC.

One obstacle against actual abuse of this construction is apparent from the above description. Only the signing CA has full control over the final contents of the P-part: an attacker will have to wait and see what serial number and validity period will be inserted. Obviously, an unpredictable P will make it impossible to concoct the collision required for a rogue certificate. On the other hand, if the full contents of P can reasonably be predicted one day in advance, nothing seems to be in the way of the construction of a rogue certificate. That, however, is not the case: the S and S' as found during the collision construction of [13] lead to RSA moduli that are too large. More precisely, S and S' both typically consist of 11 *near-collision blocks* (i.e., $11 \cdot 512$ bits) and require 5 additional blocks to generate secure 8192-bit RSA moduli. On the other hand, CAs do not necessarily accept RSA moduli of more than 2048 bits. Despite this mismatch, there was no real incentive to reduce the lengths of the RSA moduli, because the assumption that P could be predicted a day in advance sounded preposterous to begin with.

The practical validity of the above assumption came as somewhat of a surprise: practical in the sense that the prefix P cannot be predicted with 100% certainty, but with high enough probability to make further research efforts worthwhile to try and reduce the number of near-collision blocks to, say, 3. In principle the latter can be achieved by throwing more resources at the construction of the collision. It quickly turned out, as further explained below, that either the running time or the space requirements of this approach are prohibitive. To get the rogue certificate construction to work for an actual CA, a better approach to chosen-prefix collisions for MD5 was imperative.

Our improved chosen-prefix collision construction for MD5 is based on two main ingredients. In the first place, we managed to generalize the known differential path constructions (as described in [13] and extended in [12]) to an entire family of differential paths. As a result, more bits can be eliminated per pair of near-collision blocks, at a somewhat higher complexity of the actual construction of those blocks than before. This is described in Section 3, after notation and MD5 have been introduced in Section 2. The reader is forewarned that full appreciation of the improved differential paths requires familiarity with [13, Section 5]. Secondly, we introduced a variable birthday search that permits a flexible choice of search space between the two extremes of 96 bits (as in [13]) and 64 bits (as introduced in [12] and actually used for [14]): in this way more time can be invested in the birthday search to achieve a lower average number of required near-collision blocks. The details along with the more contrived parameter selection that this all leads to can be found in Section 4. The construction of the rogue CA certificate is described in Section 5. Section 6 describes an improvement creating a chosen-prefix collision using only a single near-collision block.

2 Preliminaries

2.1 Notation

MD5 operates on 32-bit words $(v_{31}v_{30}\ldots v_0)$ with $v_i \in \{0,1\}$, that are identified with elements $v = \sum_{i=0}^{31} v_i 2^i$ of $\mathbb{Z}/2^{32}\mathbb{Z}$ and referred to as 32-bit integers. In this paper we switch freely between these representations.

Integers are denoted in hexadecimal as, for instance, $\texttt{1E}_{16}$ and in binary as $\texttt{00011110}_2$. For 32-bit words X and Y we denote their bitwise AND, OR and XOR as $X \wedge Y$, $X \vee Y$ and $X \oplus Y$, respectively, $\overline{X}$ is the bitwise complement of X, the i-th bit v_i of $X = (v_{31}v_{30}\ldots v_0)$ is denoted $X[i]$, and $RL(X,n)$ (resp. $RR(X,n)$) is the cyclic left (resp. right) rotation of X by n bit positions.

For chosen message prefixes P and P' we seek suffixes S and S' such that the messages $P\|S$ and $P'\|S'$ collide under MD5. In this paper any variable X related to the message $P\|S$ or its MD5 calculation, may have a corresponding variable X' related to the message $P'\|S'$ or its MD5 calculation. Furthermore, $\delta X = X' - X$ for such a 'matched' $X \in \mathbb{Z}/2^{32}\mathbb{Z}$. For a 'matched' variable Z that consist of tuples of 32-bit integers, say $Z = (z_1, z_2, \ldots)$, we define δZ as $(\delta z_1, \delta z_2, \ldots)$.

2.2 MD5 Overview

MD5 works as follows:

1. *Padding.* Pad the message with: first a '$\texttt{1}$'-bit, next the least number of '$\texttt{0}$' bits to make the length equal to $448 \bmod 512$, and finally the bitlength of the original unpadded message as a 64-bit little-endian integer. As a result the total bitlength of the padded message is $512N$ for a positive integer N.
2. *Partitioning.* Partition the padded message into N consecutive 512-bit blocks M_1, M_2, $\ldots$, M_N.
3. *Processing.* MD5 goes through $N+1$ states IHV_i, for $0 \le i \le N$, called the *intermediate hash values* and denoted this way to achieve consistency with [13]. Each intermediate hash value IHV_i consists of four 32-bit words a_i, b_i, c_i, d_i. For $i = 0$ these are fixed public values $(a_0, b_0, c_0, d_0) = (\texttt{67452301}_{16}, \texttt{EFCDAB89}_{16}, \texttt{98BADCFE}_{16}, \texttt{10325476}_{16})$. For $i = 1, 2, \ldots, N$ intermediate hash value IHV_i is computed as $\mathrm{MD5Compress}(\mathrm{IHV}_{i-1}, M_i)$ using the MD5 compression function described in detail below.
4. *Output.* The resulting hash value is the last intermediate hash value IHV_N, expressed as the concatenation of the hexadecimal byte strings of the four words a_N, b_N, c_N, d_N, converted back from their little-endian representation.

2.3 MD5 Compression Function

The input for the compression function $\mathrm{MD5Compress}(\mathrm{IHV}, B)$ is an intermediate hash value $\mathrm{IHV} = (a, b, c, d)$ and a 512-bit message block B. The compression function consists of 64 *steps* (numbered 0 to 63), split into four consecutive *rounds* of 16 steps each. Each step t uses modular additions, a left rotation, and a non-linear function f_t. These functions involve *Addition Constants* $AC_t = \lfloor 2^{32} |\sin(t+1)| \rfloor$ for $0 \leq t < 64$, and *Rotation Constants* RC_t defined as

$$(RC_t, RC_{t+1}, RC_{t+2}, RC_{t+3}) = \begin{cases} (7,12,17,22) & \text{for } t = 0,4,8,12, \\ (5,9,14,20) & \text{for } t = 16,20,24,28, \\ (4,11,16,23) & \text{for } t = 32,36,40,44, \\ (6,10,15,21) & \text{for } t = 48,52,56,60. \end{cases}$$

The non-linear function f_t depends on the round:

$$f_t(X,Y,Z) = \begin{cases} F(X,Y,Z) = (X \wedge Y) \oplus (\overline{X} \wedge Z) & \text{for } 0 \leq t < 16, \\ G(X,Y,Z) = (Z \wedge X) \oplus (\overline{Z} \wedge Y) & \text{for } 16 \leq t < 32, \\ H(X,Y,Z) = X \oplus Y \oplus Z & \text{for } 32 \leq t < 48, \\ I(X,Y,Z) = Y \oplus (X \vee \overline{Z}) & \text{for } 48 \leq t < 64. \end{cases}$$

The message block B is partitioned into sixteen consecutive 32-bit words m_0, m_1, ..., m_{15} (with little-endian byte ordering), and expanded to 64 words W_t, for $0 \leq t < 64$, of 32 bits each:

$$W_t = \begin{cases} m_t & \text{for } 0 \leq t < 16, \\ m_{(1+5t) \bmod 16} & \text{for } 16 \leq t < 32, \\ m_{(5+3t) \bmod 16} & \text{for } 32 \leq t < 48, \\ m_{(7t) \bmod 16} & \text{for } 48 \leq t < 64. \end{cases}$$

To facilitate the analysis we follow an 'unrolled' description instead of a cyclic state. For each step t the compression function algorithm maintains a working register with 4 state words Q_t, Q_{t-1}, Q_{t-2} and Q_{t-3} and calculates a new state word Q_{t+1}. With $(Q_0, Q_{-1}, Q_{-2}, Q_{-3}) = (b, c, d, a)$, for $t = 0, 1, \ldots, 63$ in succession Q_{t+1} is calculated as follows:

$$\begin{cases} F_t &= f_t(Q_t,\ Q_{t-1},\ Q_{t-2}), \\ T_t &= F_t + Q_{t-3} + AC_t + W_t, \\ R_t &= RL(T_t, RC_t), \\ Q_{t+1} &= Q_t + R_t. \end{cases}$$

After all steps are computed, the resulting state words are added to the intermediate hash value and returned as output:

$$\text{MD5Compress}(\text{IHV}, B) = (a + Q_{61},\ b + Q_{64},\ c + Q_{63},\ d + Q_{62}). \tag{1}$$

3 A New Family of Differential Paths

The suffixes S and S' in a chosen-prefix collision consist of three consecutive parts: padding bitstrings, birthday bitstrings and near-collision bitstrings. The

Table 1. Family of partial differential paths using $\delta m_{11} = \pm 2^{q-10 \bmod 32}$

t	δQ_t	δF_t	δW_t	δT_t	δR_t	RC_t
$35-60$	0	0	0	0	0	$\cdot$
61	0	0	$\pm 2^{q-10 \bmod 32}$	$\pm 2^{q-10 \bmod 32}$	$\pm 2^{q}$	10
62	$\pm 2^{q}$	0	0	0	0	15
63	$\pm 2^{q}$	0	0	0	0	21
64	$\pm 2^{q} + \sum_{\lambda=0}^{w'} s_\lambda 2^{q+21+\lambda \bmod 32}$					

Here $w' = \min(w, 31-q)$ and $s_0, \ldots, s_{w'} \in \{-1, 0, +1\}$ for a fixed parameter $w \geq 0$. Interesting values for w are between 2 and 5.

padding bitstrings are arbitrarily chosen such that the birthday bitstrings end on the same 512-bit block border. The birthday bitstrings result in a δIHV that will be eliminated by a sequence of near-collision blocks which make up the near-collision bitstrings as described in [13, Section 5.3]. Fewer near-collision blocks are required if the family of differential paths is more effective, whereas finding a δIHV that requires fewer near-collision blocks increases the birthday search complexity. Thus, if both search time and number of near-collision blocks are limited, a more effective family of differential paths is required.

In our target application, generating a rogue CA certificate, we have to deal with two hard limits. Because the CA that is supposed to sign our (legitimate) certificate does not accept certification requests for RSA moduli larger than 2048 bits, each of our suffixes S and S' and their common appendage T must fit in 2048 bits. This implies that we can use at most 3 near-collision blocks. Furthermore, to reliably predict the serial number, the entire construction must be performed within a few days.

Thus, as shown in Table 1, we have extended the family of differential paths used to construct chosen-prefix collisions. The larger choice is parameterized by the non-negative integer w: a larger value allows elimination of more differences in δIHV per near-collision block, but increases the cost of constructing each near-collision block by a factor of roughly 2^{2w}. The value for w in Table 1 can be chosen freely, however due to the blow-up factor of 2^{2w} only the values 2, 3, 4, and 5 are of interest.

Compared to the earlier differential paths in [13, Table 2] and [12, Table 7-2], the new ones vary the carry propagations in the last 3 steps and the boolean function difference in the last step. This change affects the working state only in difference δQ_{64}. Each possible value δQ_{64} may be caused by many different carry propagations and boolean function differences. When performing the collision finding for an actual near-collision block using a particular differential path, we do not consider just one such possible variation but for the last 3 steps check only if the δQ_t's are as specified.

4 Variable Birthday Search Space, Time-Memory Trade-Off

A birthday search on a search space V is generally performed by iterating a properly chosen deterministic function $f : V \to V$ and by assuming that the points of V thus visited form a 'random walk' [9]. After approximately $\sqrt{\pi|V|/2}$ iterations one may expect to have encountered a collision, i.e., different points x and y such that $f(x) = f(y)$. Because the entire trail can in practice not be stored and to take advantage of parallelism, different pseudo-random walks are generated, of which only the startpoints, lengths, and endpoints are kept. The walks are generated to end on 'distinguished points', points with an easily recognizable bitpattern depending on $|V|$, available storage and other characteristics. The average length of a walk is inversely proportional to the fraction of distinguished points in V. Because intersecting walks share their endpoints, they can easily be detected. The collision point can then be recomputed given the startpoints and lengths of the two colliding walks.

Let p be the probability that a birthday collision satisfies additional conditions that cannot be captured by V or f. On average $1/p$ birthday collisions have to be found at a cost of $C_{\mathrm{tr}} = \sqrt{\pi|V|/(2p)}$ iterations, plus recomputation of $1/p$ intersecting walks at C_{coll} iterations. To achieve $C_{\mathrm{coll}} \approx \epsilon \cdot C_{\mathrm{tr}}$ for any given $\epsilon \leq 1$ and optimizing for the expected walk lengths, one needs to store approximately $1/(p \cdot \epsilon)$ walks. The value for p depends in an intricate way on k (cf. below), w, and the targeted number of near-collision blocks and is extensively tabulated in the final version [15] of [13]. The value for ϵ depends on the amount of available space to store walks. For very small ϵ the overall birthdaying complexity is about C_{tr}.

The first chosen-prefix collision example from [13] used a 96-bit birthday search space V with $|V| = 2^{96}$ to find a $\delta\mathrm{IHV} = (\delta a, \delta b, \delta c, \delta d)$ with $\delta a = 0$, $\delta b = \delta c = \delta d$. This search can be continued until a birthday collision is found that requires a sufficiently small number of near-collision blocks, which leads to a trade-off between the birthday search and the number of blocks. If one would aim for just 3 near-collision blocks, one expects $2^{57.33}$ MD5 compressions for the 96-bit birthday search, which would take about 50 days on 215 PlayStation 3 game consoles.

By leaving δb free, we get an improved 64-bit search space (cf. [12], [14]). In the resulting birthday collisions, the differences in δb compared to δc were handled by the differential path from [12, section 7.4] which corresponds to $\delta Q_{64} = \pm 2^{q} \mp 2^{q+21 \bmod 32}$ in Table 1 (cf. equation 2.3(1)). This significantly decreasing the birthday search complexity, but also increases the average number of near-collision blocks. When aiming for 3 blocks, birthdaying requires about $2^{55.73}$ MD5 compressions. But the probability that a birthday collision is useful becomes so small that the space requirements are prohibitive: about $2^{50.15}$ bytes, i.e., more than a petabyte.

A more flexible approach is obtained by interpolating between the above 64-bit and 96-bit birthday searches, while exploiting the family of differential paths from Section 3. For any $k \in \{0, 1, \ldots, 32\}$, we can do a $(64 + k)$-bit search

Table 2. Birthday complexities and memory requirements for $k = 0$

	$w=0$		$w=1$		$w=2$		$w=3$		$w=4$		$w=5$	
r	C_{tr}	M	C_{tr}	M	C_{tr}	M	C_{tr}	M	C_{tr}	M	C_{tr}	M
14	$2^{36.68}$	1MB	$2^{34.01}$	1MB	$2^{32.96}$	1MB	$2^{32.84}$	1MB	$2^{32.83}$	1MB	$2^{32.83}$	1MB
13	$2^{37.55}$	1MB	$2^{34.69}$	1MB	$2^{33.22}$	1MB	$2^{32.93}$	1MB	$2^{32.88}$	1MB	$2^{32.87}$	1MB
12	$2^{38.55}$	1MB	$2^{35.59}$	1MB	$2^{33.71}$	1MB	$2^{33.16}$	1MB	$2^{33.02}$	1MB	$2^{32.98}$	1MB
11	$2^{39.68}$	2MB	$2^{36.71}$	1MB	$2^{34.50}$	1MB	$2^{33.63}$	1MB	$2^{33.34}$	1MB	$2^{33.24}$	1MB
10	$2^{40.97}$	11MB	$2^{38.06}$	1MB	$2^{35.60}$	1MB	$2^{34.42}$	1MB	$2^{33.91}$	1MB	$2^{33.71}$	1MB
9	$2^{42.40}$	79MB	$2^{39.63}$	2MB	$2^{37.02}$	1MB	$2^{35.56}$	1MB	$2^{34.80}$	1MB	$2^{34.45}$	1MB
8	$2^{44.02}$	732MB	$2^{41.43}$	21MB	$2^{38.76}$	1MB	$2^{37.09}$	1MB	$2^{36.05}$	1MB	$2^{35.51}$	1MB
7	$2^{45.73}$	8GB	$2^{43.43}$	323MB	$2^{40.83}$	9MB	$2^{39.02}$	1MB	$2^{37.73}$	1MB	$2^{36.95}$	1MB
6	$2^{47.92}$	164GB	$2^{45.69}$	7GB	$2^{43.22}$	241MB	$2^{41.40}$	20MB	$2^{39.89}$	3MB	$2^{38.85}$	1MB
5	$2^{49.82}$	3TB	$2^{47.92}$	164GB	$2^{45.89}$	10GB	$2^{44.20}$	938MB	$2^{42.59}$	102MB	$2^{41.34}$	18MB
4					$2^{49.33}$	2TB	$2^{47.42}$	82GB	$2^{45.81}$	9GB	$2^{44.55}$	2GB
3											$2^{48.17}$	231GB

similar to the one above, but with $\delta b = \delta c \bmod 2^k$. Since δb does not introduce new differences compared to δc in the lower k bits, the average number of near-collision blocks may be reduced – in particular when taking advantage of our new family of differential paths – while incurring a higher birthdaying cost. For any targeted number of near-collision blocks, this leads to a trade-off between the birthdaying cost and space requirements (unless the number of blocks is at least 6, since then 241MB suffices for the plausible choice $w = 2$). Table 2 gives birthday complexities for $k = 0$, a range of w-values to control the number of differences that can be eliminated per near-collision block, and number r of near-collision blocks. The smallest amount of memory required for C_{coll} to be smaller than C_{tr} is denoted by M.

Having a cluster of 215 PlayStation 3 (PS3) game consoles at our disposal obviously influenced our parameter choices. When running Linux on a PS3, our application has access to 6 Synergistic Processing Units (SPUs), a general purpose CPU, and about 150MB of RAM per PS3. For our birthday search, the 6×215 SPUs are computationally equivalent to approximately 8600 regular 32-bit cores, due to each SPU's 4×32-bit wide SIMD architecture. The other parts of the chosen-prefix collision construction are not suitable for the SPUs, but we were able to use the 215 PS3 CPUs for the construction of the actual near-collision blocks. With these resources, the choice $w = 5$ still turned out to be acceptable despite the 1000-fold increase in the cost of the actual near-collision block construction. This is the case even for the hard cases with many differences between IHV and IHV$'$: as a consequence the differential paths contain many bitconditions which leaves little space for so-called 'tunnels' (cf. [6]), thereby complicating the near-collision block construction.

For $w = 5$ and the targeted 3 near-collision blocks, Table 3 shows the time-memory tradeoff when the birthday search space is varied with k. With 150MB at our disposal per PS3, for a total of about 30GB, we decided to use $k = 8$ as this optimizes the overall birthday complexity for the plausible case that the birthday

Table 3. Birthday complexities and memory requirements for $r = 3$

	$w = 3$		$w = 4$		$w = 5$	
k	C_{tr}	M	C_{tr}	M	C_{tr}	M
0					$2^{48.17}$	231GB
2					$2^{49.10}$	210GB
4			$2^{50.43}$	330GB	$2^{49.29}$	68GB
6	$2^{51.33}$	287GB	$2^{50.54}$	96GB	$2^{49.69}$	30GB
8	$2^{51.98}$	177GB	$2^{50.74}$	32GB	$2^{49.99}$	11GB
10	$2^{52.43}$	82GB	$2^{51.24}$	16GB	$2^{50.44}$	5GB
12	$2^{52.44}$	22GB	$2^{51.64}$	7GB	$2^{50.90}$	3GB
14	$2^{52.76}$	9GB	$2^{52.01}$	3GB	$2^{51.38}$	2GB
16	$2^{53.13}$	4GB	$2^{52.48}$	2GB	$2^{51.96}$	675MB
18	$2^{53.59}$	2GB	$2^{53.02}$	733MB	$2^{52.61}$	418MB
20	$2^{53.96}$	673MB	$2^{53.46}$	340MB	$2^{53.13}$	215MB
22	$2^{54.43}$	324MB	$2^{54.01}$	182MB	$2^{53.73}$	123MB
24	$2^{54.92}$	160MB	$2^{54.59}$	102MB	$2^{54.33}$	71MB
26	$2^{55.52}$	92MB	$2^{55.25}$	64MB	$2^{55.04}$	47MB
28	$2^{56.11}$	52MB	$2^{55.95}$	42MB	$2^{55.83}$	36MB
30	$2^{56.74}$	32MB	$2^{56.68}$	29MB	$2^{56.61}$	26MB
32	$2^{57.27}$	17MB	$2^{57.27}$	17MB	$2^{57.27}$	17MB

search takes $\sqrt{2}$ times longer than expected. The overall chosen-prefix collision construction takes on average less than a day on the cluster of PS3s. In theory we could have used 1TB (or more) of hard drive space, in which case it would have been optimal to use $k = 0$ for a birthday search of about 20 PS3 days.

5 Rogue CA Certificate Construction

In this section we present some of the details of the construction of the to-be-signed parts of our colliding certificates, as outlined in Figure 1.

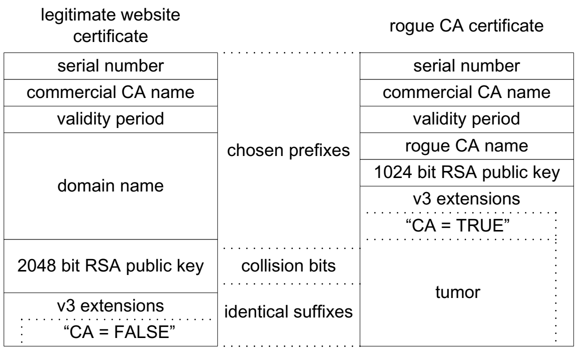


Fig. 1. The to-be-signed parts of the colliding certificates

The chosen prefix of the website certificate contains a subject Distinguished Name (a domain name), as well as the first 208 bits of the RSA modulus, chosen at random, as padding to reach proper alignment with the rogue CA certificate. Furthermore, an educated guess has to be included for the serial number and validity period fields that the signing CA will insert when it processes the legitimate website's certification request. For the targeted commercial CA it turned out, based on repeated observations, that the validity period can be predicted very reliably as the period of precisely one year plus one day, starting exactly six seconds after a certification request is submitted. Furthermore, it was found that the targeted CA uses sequential serial numbers. Being able to predict the next serial number, however, is not enough, because the construction of the collision can be expected to take at least a day, implying a substantial and uncertain increment in the serial number by the time the collision construction is finished. The increment in serial number over a weekend, however, does not vary a lot and Monday morning's serial numbers can be predicted, roughly, on the Friday afternoon before.

The chosen prefix of the rogue CA certificate contains a short rogue CA name, a 1024-bit RSA public key, and the first part of the X.509v3 extension fields. One of these extension fields is the 'basic constraints' field, containing a bit that identifies the certificate as a CA certificate (in Figure 1 denoted by "CA=TRUE"). The final part of the rogue chosen prefix contains an indication that all remaining bits of this to-be-signed part should be interpreted as an extension field of the type "Netscape Comment", a field that is ignored by most application software. In Figure 1 this field is denoted as 'tumor'.

Given these two chosen prefixes, the collision bits consisting of birthday bits and near-collision blocks are computed as described above. We describe how those bits are interpreted on either side. The birthday bits occupy 96 bits. Immediately after them there is a border between MD5 input blocks. In the website certificate the birthday bits are part of the RSA modulus, in the rogue CA certificate they belong to the tumor.

After the birthday bits, there are 3 near-collision blocks of 512 bits each. In the website certificate these are part of the RSA modulus, thereby fixing $208 + 96 + 3 \times 512 = 1840$ bits of the website's RSA modulus. In the rogue CA certificate these 3 blocks are the second part of the tumor.

After the collision bits, another $2048 - 1840 = 208$ bits are needed to complete the website's 2048-bit RSA modulus. These 208 bits have to be determined in such a way that the complete factorization of the RSA modulus is known, in order to be able to submit a valid certificate signing request for the website. The RSA modulus does not have to be secure as it will not be used after obtaining the website's certificate. Its least significant 208 bits are determined as follows. Let B denote the fixed 1840-bit part of the RSA modulus followed by 208 one bits. Now select a random 224-bit integer q until $B \bmod q$ is less than 2^{208}, and keep doing so until both q and $\lfloor B/q \rfloor$ are prime. As a result $n = \lfloor B/q \rfloor q$ has the desired 1840 leading bits and, for purely esthetic reasons, n's smallest prime factor q is larger than the 67-digit largest factor found (so far) using the Elliptic Curve integer factorization method.

Finally the website's RSA public exponent is set, followed by the X.509v3 extensions of the website certificate. All bits after the collision bits in the website certificate's to-be-signed part are copied to the tumor in the rogue CA certificate.

A legitimate PKCS#10 Certificate Signing Request can now be submitted to the signing CA. This CA requires proof of possession of the private key corresponding to the public key inside the request. This is done by signing the request using this private key and this is the sole reason that we needed the factorization of the website's RSA modulus. Upon correct submission, the signing CA returns a website certificate. If the serial number and validity period as inserted by the CA indeed match our guess, then the website certificate's to-be-signed part will collide under MD5 with the rogue CA certificate's to-be-signed part, and the signing CA's MD5-based digital signature will be equally valid for the rogue data.

Getting the right serial number at the right time requires some care. About half an hour before the targeted submission moment, the same request is submitted, and the serial number in the resulting certificate is inspected. If it is already too high, the entire attempt has to be abandoned. Otherwise, the request is repeatedly submitted, with a frequency depending on the gap that may still exist between the serial number received and the targeted one, and taking into account possible certification requests by others. In this way the serial number is slowly nudged toward the right value at the right time.

A proof of concept rogue CA certificate constructed in this manner, where it required some experimentation and a moderate number of attempts to get the correct serial number and validity period, was obtained using a commercial CA. Full details, including the rogue CA certificate, are available from `www.win.tue.nl/hashclash/rogue-ca/`.

6 Independent Additional Improvement

We show how to construct a chosen-prefix collision for MD5 that consists of 84 birthday bits followed by one pair of near-collision blocks, for a chosen-prefix collision-causing appendage of $84 + 512 = 596$ bits. The construction is based on an even richer family of differential paths that allows elimination using a single pair of near-collision blocks of a set of δIHVs that is bounded enough so that finding the near-collision blocks is still feasible, but large enough that such a δIHV can be found efficiently by a birthday search. Instead of using the family of differential paths based on $\delta m_{11} = \pm 2^i$, we use the fastest known collision attack for MD5 and vary the last few steps to find a large family of differential paths.

We first present a new collision attack for MD5 with complexity of approximately 2^{16} MD5 compressions improving upon the $2^{20.96}$ MD5 compressions required in [20]. Our starting point is the partial differential path for MD5 given in Table 4. It is based on message differences $\delta m_2 = 2^8$, $\delta m_4 = \delta m_{14} = 2^{31}$ and

Table 4. Partial differential path for fast near-collision attack

t	δQ_t	δF_t	δW_t	δT_t	δR_t	RC_t
$30-33$	0	0	0	0	0	$\cdot$
34	0	0	2^{15}	2^{15}	2^{31}	16
35	2^{31}	2^{31}	2^{31}	0	0	23
36	2^{31}	0	0	0	0	4
37	2^{31}	2^{31}	2^{31}	0	0	11
$38-46$	2^{31}	2^{31}	0	0	0	$\cdot$
47	2^{31}	2^{31}	2^{8}	2^{8}	2^{31}	23
48	0	0	0	0	0	6
49	0	0	0	0	0	10
50	0	0	2^{31}	0	0	15
$51-59$	0	0	0	0	0	$\cdot$
60	0	0	2^{31}	2^{31}	-2^{5}	6
61	-2^{5}	0	2^{15}	2^{15}	2^{25}	10
62	$-2^{5}+2^{25}$	0	2^{8}	2^{8}	2^{23}	15
63	$-2^{5}+2^{25}+2^{23}$	$2^{5}-2^{23}$	0	$2^{5}-2^{23}$	$2^{26}-2^{14}$	21
64	$-2^{5}+2^{25}+2^{23}+2^{26}-2^{14}$					

Partial differential path for $t = 29, \ldots, 63$ using message differences $\delta m_2 = 2^8$, $\delta m_4 = \delta m_{14} = 2^{31}$, $\delta m_{11} = 2^{15}$. The probability that it is satisfied is approximately $2^{-14.5}$.

$\delta m_{11} = 2^{15}$ which is very similar to those used by Wang et al. in [17] for the first collision attack against MD5. This partial differential path can be used for a near-collision attack with complexity of approximately $2^{14.8}$ MD5 compressions. This leads in the usual fashion to an identical-prefix collision attack for MD5 that requires approximately 2^{16} MD5 compressions, since one has to do it twice: first to add differences to δIHV and then to eliminate them again. It should be noted that usually bitconditions are required on the IHV and IHV$'$ between the two collision blocks which imply an extra factor in complexity. In the present case, however, we can construct a large set of differential paths for the second near-collision block that will cover all bitconditions that are likely to occur, thereby avoiding the extra complexity.

By properly tuning the birthday search, the same partial differential path leads to the construction of a single near-collision block chosen-prefix collision for MD5. By varying the last steps of the differential path and by allowing the collision finding complexity to grow by a factor of about 2^{26}, we have found a set $\mathcal{S}$ of about $2^{23.3}$ different $\delta\mathrm{IHV} = (\delta a, \delta b, \delta c, \delta d)$ of the form $\delta a = -2^5$, $\delta d = -2^5 + 2^{25}$, $\delta c = -2^5 \bmod 2^{20}$ that can be eliminated. Such δIHVs can be found using an 84-bit birthday search with step function $f : \{0,1\}^{84} \to \{0,1\}^{84}$ of the form

$$f(x) = \begin{cases} \phi(\mathrm{MD5compress}(\mathrm{IHV}, B\|x) + \widehat{\delta\mathrm{IHV}}) & \text{for } \sigma(x) = 0 \\ \phi(\mathrm{MD5compress}(\mathrm{IHV}', B'\|x)) & \text{for } \sigma(x) = 1, \end{cases}$$

Table 5. Example single-block chosen-prefix collision

Message 1
4F64656420476F6C6472656963680A4F64656420476F6C6472656963680A4F64
656420476F6C6472656963680A4F64656420476FD8050D0019BB9318924CAA96
DCE35CB835B349E144E98C50C22CF461244A4064BF1AFAECC5820D428AD38D6B
EC89A5AD51E29063DD79B16CF67C12978647F5AF123DE3ACF844085CD025B956
Message 2
4E65616C204B6F626C69747A0A4E65616C204B6F626C69747A0A4E65616C204B
6F626C69747A0A4E65616C204B6F626C69747A0A75B80E0035F3D2C909AF1BAD
DCE35CB835B349E144E88C50C22CF461244A40E4BF1AFAECC5820D428AD38D6B
EC89A5AD51E29063DD79B16CF6FC11978647F5AF123DE3ACF84408DCD025B956

where $\delta\widehat{\mathrm{IHV}}$ is of the required form, $\sigma : x \mapsto \{0, 1\}$ is a balanced selector function and $\phi(a, b, c, d) \mapsto a\|d\|(c \bmod 2^{20})$. There are $2^{128-84} = 2^{44}$ possible δIHVs of this form, of which only about $2^{23.3}$ are in the allowed set $\mathcal{S}$. It follows that a birthday collision has probability $p = 2^{23.3}/(2^{44} \cdot 2) = 2^{-21.7}$ to be useful, where the additional factor 2 stems from the fact that different prefixes are required.

A useful birthday collision can be expected after $\sqrt{\pi 2^{84}/(2p)} \approx 2^{53.2}$ MD5 compressions, requires 400MB of storage and takes about 3 days on 215 PS3s. The expected complexity of finding the actual near-collision blocks is bounded by about $2^{14.8+26} = 2^{40.8}$ MD5 compressions. In Table 5 two 128-byte messages are given both consisting of a 52-byte chosen prefix and a 76-byte single-block chosen-prefix collision suffix and with colliding MD5 hash value `D320B6433D8EBC1AC65711705721C2E1`.

7 Conclusion

We have shown that the length of formerly rather long chosen-prefix collisions for MD5 can be reduced to a minimum at a still acceptable cost, and that short enough chosen-prefix collision-causing appendages can be found fast enough to cause trouble, if so desired.

As secure cryptographic hash function for digital signature applications, MD5 has been declared dead over and over again. The improvements in the collision construction for MD5 presented here firmly hammer another nail into its coffin. We have been told that simply removing all existing MD5 applications would break too much. Nevertheless, we hope that our work has contributed to a sooner ending of MD5's funeral.

In Table 6 we present a historical overview of the decline in complexity of MD5 and SHA-1 collision finding. It clearly illustrates that attacks only get better, not worse. Not reflected in the table is the fact that already in 1993 it was known that there was serious trouble with MD5, based on collisions in its compression function (cf. [1], [3]). We leave any speculation about the future of SHA-1 cryptanalysis to the knowledgeable reader.

A possible mitigation of the risk posed by chosen-prefix collisions when signing documents is to let the signer add a sufficient amount of fresh randomness at

Table 6. Collision complexities – Historical overview

	MD5		SHA-1	
year	identical-prefix	chosen-prefix	identical-prefix	chosen-prefix
pre-2004	2^{64} (trivial)	2^{64} (trivial)	2^{80} (trivial)	2^{80} (trivial)
2004	2^{40} [16], [17]			
2005	2^{37} [5]		2^{69} [18]	
			2^{63} [19]	
2006	2^{32} [6], [11]	2^{49} [13]		$2^{80-\epsilon}$ [10]
2007	2^{25} [12]	2^{42} [12]	2^{61} [8]	
2008	2^{21} [20]			
2009	2^{16} (this paper)	2^{39} (this paper)	2^{52} [7]	

Complexity is given as the number of calls to the relevant compression function. The figures are optimized for speed, i.e., for collisions using any number of near-collision blocks. For other collision lengths the complexities may differ.

the appropriate spot in the to-be-signed data, i.e., not as a suffix but preferably somewhere early on. For certificates the serial number, or even a somewhat variable validity period, would be an appropriate spot. Although this would work, it can be argued that such a countermeasure relies on unintentional choices of the X.509 certificate standard. Indeed, we would be in favor of a more fundamental way to add randomness to to-be-hashed data, such as using randomized hashing as a mode of operation for hash functions as proposed in [4]. The collision was, at least partially, achievable because of 'flabby structure' of the certificate (cf. [2]), so that may have to be addressed as well. On the other hand, a more 'rigid' structure would not be an excuse to use a poor hash function: irrespective of the elegance or lack thereof of the certificate structure, we need a solid hash function.

As far as we know, no harm was done using our rogue CA certificate. The positive effects we intended to achieve by its construction have been realized. From this point of view, and because it required new cryptanalytic insights in MD5, the project described in this paper was very gratifying. Nevertheless, there was another, secondary aspect that is worth mentioning here. Although, as stated earlier, creating havoc was not our goal, we must admit that some havoc was created by our announcement. Despite our best efforts to inform the party that was arguably most directly affected by our work (as documented on one of the related websites), we also felt we should not reveal our identities to avoid any attempt to file an injunction barring our announcement. Overall, this did not stimulate a healthy exchange of information of which all parties involved could have profited. We do not know how the present legal climate could best be changed to address this problem, but hope that the difficulties as clearly observed in our case help to expedite a solution.

Acknowledgements

We are grateful for comments by the Crypto 2009 reviewers, and support by the European Commission through the EU ICT program ECRYPT II, by the Swiss National Science Foundation, and by EPFL DIT.

References

1. den Boer, B., Bosselaers, A.: Collisions for the compression function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
2. Diffie, W.: Personal communication (January 2009)
3. Dobbertin, H.: Cryptanalysis of MD5 Compress (May 1996), `http://www-cse.ucsd.edu/~bsy/dobbertin.ps`
4. Halevi, S., Krawczyk, H.: Strengthening Digital Signatures via Randomized Hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006), `http://tools.ietf.org/html/draft-irtf-cfrg-rhash-01`
5. Klima, V.: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications, Cryptology ePrint Archive, Report 2005/102
6. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute, Cryptology ePrint Archive, Report 2006/105
7. McDonald, C., Hawkes, P., Pieprzyk, J.: SHA-1 collisions now 2^{52}. In: Eurocrypt 2009 Rump session
8. Mendel, F., Rechberger, C., Rijmen, V.: Update on SHA-1. In: Crypto 2007 Rump session
9. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology 12(1), 1–28 (1999)
10. Rechberger, C.: Unpublished result (2006)
11. Stevens, M.: Fast Collision Attack on MD5, Cryptology ePrint Archive, Report 2006/104
12. Stevens, M.: On collisions for MD5, Master's thesis, TU Eindhoven (June 2007), `http://www.win.tue.nl/hashclash/`
13. Stevens, M., Lenstra, A., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
14. Stevens, M., Lenstra, A., de Weger, B.: Predicting the winner of the 2008 US presidential elections using a Sony PlayStation 3 (2007), `http://www.win.tue.nl/hashclash/Nostradamus/`
15. Stevens, M., Lenstra, A., de Weger, B.: Chosen-Prefix Collisions for MD5 and Applications (in preparation)
16. Wang, X., Lai, X., Feng, D., Yu, H.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. In: Crypto 2004 Rump Session (2004)
17. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
18. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
19. Wang, X., Yao, A., Yao, F.: New Collision Search for SHA-1. In: Crypto 2005 Rump session
20. Xie, T., Liu, F., Feng, D.: Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5?, Cryptology ePrint Archive, Report 2008/391

Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1

Kazumaro Aoki and Yu Sasaki

NTT, 3-9-11 Midoricho, Musashino-shi, Tokyo 180-8585 Japan

Abstract. Preimage resistance of several hash functions has already been broken by the meet-in-the-middle attacks and they utilize a property that their message schedules consist of only permutations of message words. It is unclear whether this type of attacks is applicable to a hash function whose message schedule does not consist of permutations of message words. This paper proposes new attacks against reduced SHA-0 and SHA-1 hash functions by analyzing a message schedule that does not consist of permutations but linear combinations of message words. The newly developed cryptanalytic techniques enable the meet-in-the-middle attack to be applied to reduced SHA-0 and SHA-1 hash functions. The attacks find preimages of SHA-0 and SHA-1 in $2^{156.6}$ and $2^{159.3}$ compression function computations up to 52 and 48 steps, respectively, compared to the brute-force attack, which requires 2^{160} compression function computations. The previous best attacks find preimages up to 49 and 44 steps, respectively.

Keywords: SHA-0, SHA-1, meet-in-the-middle, one-way, preimage.

1 Introduction

After the breakthrough described in Wang's study [14], much attention has been paid to the security of MD4-like hash functions such as MD4 [7], MD5 [8], HAVAL [15], and SHA family [13]. First, attention was focused on collision resistance, and the more recently, attention has been focused on preimage resistance. Preimage resistance is more important than collision resistance because the security of many applications employing hash functions are based on preimage resistance, and breaking preimage resistance implies breaking collision resistance of the practical hash functions[1], and, therefore, we should focus more attention on preimage resistance. Saarinen showed a preimage attack on new FORK-256 [4] in 2007 [9], and in 2008, Leurent showed that a preimage of MD4 can be computed to the complexity of $2^{100.51}$ MD4 computations [5]. In these attack, the meet-in-the-middle technique helps to compute the preimage. After

[1] Collision resistance implies preimage resistance for hash functions with uniformly random output. Note that collision resistance does not always imply preimage resistance. Construction of such artificial hash functions is explained in [6, Note 9.20].

this, the meet-in-the-middle attack is directly used to compute a (second) preimage of hash functions [1,2,10,11,12], and the meet-in-the-middle technique seems to be a very powerful tool to compute a preimage.

SHA-1 is a widely used hash function, and its security assessment is very important. Actually, many public key encryption and signature schemes use SHA-1 as a random oracle, and the SSL/TLS protocol uses SHA-1 for many purposes. In Crypto 2008, [3] showed the first preimage attacks against reduced SHA-0 and SHA-1. Its authors use "reversing the inversion problem" and attacked SHA-0 and SHA-1 up to 49 and 44 steps, respectively. On the other hand, the resistance of SHA-0 and SHA-1 against the meet-in-the-middle technique is an interesting problem to be studied. However, the previous preimage attacks using the meet-in-the-middle technique are only applied to hash functions whose message schedule consists of permutations of the message words, while the message schedules of SHA-0 and SHA-1 are in more complicated form, that is, linear transformation of message words. Moreover, the techniques developed in [3] seems not to be able to be applied to the framework of the meet-in-the-middle attack.

On conducting the meet-in-the-middle attack, first we partition steps for SHA-0 (or SHA-1) into two *chunks*. A chunk comprises consecutive steps of the corresponding hash function and includes at least one *neutral word*, which appears in the chunk and does not appear in the other chunk. So, steps in a chunk can be executed using the neutral word for the chunk, and does not require the neutral word for the other chunk. Although finding neutral words is important in this scenario, the previous meet-in-the-middle attack only are applied to hash functions such as MD4, which has a simple message schedule, that is, consisting only of permutations of message words. So, a message word itself can be regarded as a neutral word, and it is very easy to find chunks of long steps. For example, [10] can compute a (second) preimage of HAVAL-5 up to 151 steps. To apply the same strategy to SHA-0 or SHA-1, we face the first difficulty which is that we cannot find any neutral words for long steps, because SHA-0 and SHA-1 adopt the linear transformation of a message word as message schedule, and the linear transformation spreads the effect of a message word to many steps and prevents finding neutral words. To solve this problem, we seek chunks that the rank of their matrix representation is not full, and we regard the kernel generators of linear transformations of each chunk as neutral words. This seems to be a good idea, in fact, if message words included in the kernel generators of the first and the second chunks are different, each kernel can be computed independently, and thus, the meet-in-the-middle attack can be performed. However, we face the second difficulty that the kernel generators for two chunks may share the same message word, and how to determine the value of the neutral word in each chunk is unclear. To overcome the second problem, we convert a message schedule by multiplying by a regular matrix, so that converting a message schedule using a matrix results in converted kernel generators for two chunks becoming unit vectors. That is, we can choose converted message words as neutral words, and this enables us to apply the existing meet-in-the-middle attack to SHA-0 and SHA-1.

Table 1. Preimage Attacks Against SHA-0 and SHA-1

		[3]			Current Results		
		# of	Complexity		# of	Complexity	
	Attack Type	Steps	Time	Memory	Steps	Time	Memory
SHA-0	pseudo-preimage	50	2^{158}	2^{25}	52	$2^{151.2}$	2^{15}
					52	$2^{152.2}$	negligible
	preimage	49	2^{159}	2^{25}	52	$2^{156.6}$	2^{15}
					52	$2^{157.1}$	negligible
SHA-1	pseudo-preimage	45	2^{157}	2^{20}	48	$2^{156.7}$	2^{40}
					48	$2^{157.7}$	negligible
	preimage	44	2^{157}	2^{20}	48	$2^{159.3}$	2^{40}
					48	$2^{159.8}$	negligible

The unit of time complexity is one compression function computation, and the unit of memory complexity is a few times of the hash length which is 160 bits.

This paper presents a new analysis method for a linear message schedule, which enables us to utilize the meet-in-the-middle technique to compute a preimage effectively. We then apply this technique to SHA-0 and SHA-1. The newly developed analysis is a generalization of the previously reported analysis for MD5 and other hash functions [10,11,12]. The technique with the detailed analysis of step functions can find a preimage of reduced SHA-0 and SHA-1 faster than the brute-force attack up to 52 and 48 steps, respectively (out of 80 steps), which are the best results so far. Table 1 summarizes the preimage attacks against SHA-0 and SHA-1. We also note the complexity of memoryless attack in Table 1.

2 Preliminaries

2.1 Specification of SHA-0 and SHA-1

This paper focuses on SHA-b ($b = 0$ or 1). This section shows the specifications of SHA-b used in this paper. For more details, please refer to the original specifications [13].

SHA-b adopts the Merkle-Damgård structure [6, Algorithm 9.25]. The message string is first padded to be a 512-bit multiple, and divided into 512-bit blocks, $(M_0, M_1, \ldots, M_{m-1})$ $(M_i \in \{0,1\}^{512})$. The compression function inputs a 512-bit message string and 160-bit chaining variable. The message blocks are input to the iterative use of compression function CF to compute hash value H_m.

$$H_0 \leftarrow \mathrm{IV}, \qquad H_{i+1} \leftarrow \mathrm{CF}(H_i, M_i) \quad (i = 0, 1, \ldots, m-1)$$

where IV is the constant defined in the specification.

The compression function is based on the Davies-Meyer mode [6, Algorithm 9.42]. Let $^{\lll x}$ denote the x-bit left rotation. First, the message block is expanded using the message schedule algorithm.

$$\begin{cases} w_j \leftarrow m_j, & (\ 0 \le j < 16) \\ w_j \leftarrow (w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16})^{\lll b}, & (16 \le j < 80) \end{cases} \tag{1}$$

where $(m_0, m_1, \ldots, m_{15}) \leftarrow M_i$ ($m_j \in \{0,1\}^{32}$). Hereafter, we call a 32-bit string a *word*. Then, the step functions are applied.

$$p_0 \leftarrow H_i, \quad p_{j+1} \leftarrow R_j(p_j, w_j) \text{ (for } j = 0, 1, \ldots, 79), \quad H_{i+1} \leftarrow H_i + p_{80}, \tag{2}$$

where "+" denotes the wordwise addition. Step function R_j is defined as given hereafter:

$$\begin{cases} a_{j+1} \leftarrow a_j^{\lll 5} + f_j(b_j, c_j, d_j) + e_j + w_j + k_j \\ b_{j+1} \leftarrow a_j, \ c_{j+1} \leftarrow b_j^{\lll 30}, \ d_{j+1} \leftarrow c_j, \ e_{j+1} \leftarrow d_j \end{cases}$$

where $(a_j, b_j, c_j, d_j, e_j) = p_j$, f_j is a bitwise function, and k_j is a constant specified by the specification.

Note that the difference between SHA-0 and SHA-1 is only the existence of the rotation in Eq.(1).

2.2 Converting Pseudo-preimage Attack to Preimage Attack

We call (H_i, M_i) a *pseudo-preimage* of the compression function, where the given H_{i+1} satisfies $H_{i+1} = \mathrm{CF}(H_i, M_i)$. Hereafter, we use the computational unit as one computation of the compression function.

[6, Fact 9.99] gives an algorithm for converting a pseudo-preimage attack to a preimage attack for the Merkle-Damgård construction. A preimage can be computed in $2^{1+(x+n)/2}$ with one more block message, where the hash value is n-bit long, when a pseudo-preimage can be computed in 2^x.

Note that the attacks [3,5] generalize this conversion, tree and graph based approaches. Their conversions require to fix some part of hash value and pseudo-preimage in the pseudo-preimage attack, and to generate this combination at very small cost. Unfortunately, our attack described later cannot satisfy this condition. So, we cannot use tree and graph based approaches with our attacks.

2.3 Meet-in-the-Middle Attack

This section describes the basic strategy of the preimage attack using the meet-in-the-middle attack proposed in [1].

Assume that the message length with padding is equal to one block. The hash value is computed by $H_1 = \mathrm{CF}(\mathrm{IV}, M_0)$. Focusing on Eq.(2) reduced to s steps, we assume that some t, u, and v exist with the following conditions.

$$\begin{cases} w_j \ (0 \le j < t) \text{ is independent of } m_v \\ w_j \ (t \le j < s) \text{ is independent of } m_u \end{cases} \tag{3}$$

We can construct the following algorithm.

0. Choose m_j ($j \in \{0, 1, \ldots, 15\} \backslash \{u, v\}$) arbitrary.
1. For all $m_u \in \{0,1\}^{32}$, compute $p_t \leftarrow R_{t-1}(R_{t-2}(\cdots R_0(\mathrm{IV}, w_0) \cdots, w_{t-2}), w_{t-1})$ and store (m_u, p_t) in a table.

2. For all $m_v \in \{0,1\}^{32}$, compute $p_t \leftarrow R_t^{-1}(R_{t+1}^{-1}(\cdots R_{s-1}^{-1}(p_s, w_{s-1}) \cdots, w_{t+1}), w_t)$, where $p_s \leftarrow H_1 - \mathrm{IV}$ and "$-$" denotes the wordwise subtraction. If one of the p_ts has a match in the table generated in 1, M_0 $(= (m_0, m_1, \ldots, m_{15}))$ is a preimage of the hash function.

The complexity of the above algorithm is about 2^{32}, and the success probability is about $2^{-160+64}$. Thus, to iterate the above algorithm 2^{160-64} times, we expect to find a preimage with high probability. The time complexity of the attack is 2^{160-32} and the memory complexity is about 6×2^{32} words.

Hereafter, we call such m_u and m_v *neutral words*, and call consecutive steps $j \in [0, t)$ and $j \in [t, s)$ *chunks*. In this meet-in-the-middle attack, how to find two chunks with a neutral word is important. Section 3 describes how to find this that satisfies Condition (3) with given w_i $(i = 0, 1, \ldots, s-1)$.

2.4 Auxiliary Techniques with the Meet-in-the-Middle Attack

This section describes the techniques proposed in [1,11] that can be used with the algorithm described in Section 2.3. These techniques improve the complexity and increase the number of steps that can be attacked by the attack described in Section 2.3.

Splice-and-cut. The meet-in-the-middle attack in Section 2.3 starts to compute input p_0 in step 0 and output p_s in step $s-1$. Considering the final addition in the Davies-Meyer mode in Eq.(2), we regard that the final and the first steps are consecutive. Thus, we can determine that the meet-in-the-middle attack starts with any step and matches with any step. We call this technique *splice-and-cut* [1]. Note that this technique will produce a *pseudo*-preimage, because IV cannot be controlled by an attacker, though we can compute a preimage using Section 2.2.

Partial-matching and partial-fixing. The step function R_j in SHA-b does not update all words in p_j. In fact, all words in p_j match p_{j+1} except one word. This fact enables us not to fix matching-step t in Section 2.3, and Condition (3) changes from the chunk partition of $[0, t)$ and $[t, s)$ to that of $[0, t)$ and $[t+c, s)$, where $c \leq 4$. This may increase the number of steps that a preimage can be computed because we may be able to include the neutral words m_u and m_v in the steps $[t, t+c)$. This loosens the conditions based on which the neutral words are selected and how the chunks are selected. We call this technique *partial-matching* [1].

Moreover, we can choose a larger c than that for partial-matching, by fixing partial bits in m_u and/or m_v, since the partial bits in p_j depending on m_u or m_v $(j \in [t, t+c))$ can be computed. We call this technique *partial-fixing* [1]. In the case of SHA-b, with manual attempts, c seems to be chosen up to ≈ 15. An example of partial-matching with partial-fixing is provided in a later section.

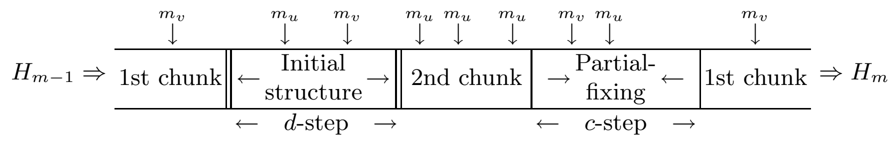


Fig. 1. A chunk partition with initial structure and partial-fixing technique

Initial structure. In the partial-matching or partial-fixing technique, we can ignore several steps regarding neutral words for the matching-part in the meet-in-the-middle attack to choose the appropriate chunks. Similarly, we can ignore several steps for the starting-part in the meet-in-the-middle attack. A preliminary version of the technique is introduced in [2], and it can be considered as a local collision [10] similar to existing collision attacks. Using the local collision technique, neutral words should be chosen at the edges of the starting-part. After computing the matching-part, we should confirm that the values of the neutral words satisfies the condition of the local collision. This condition is satisfied with probability 2^{-32}, and we lose the advantage to use the meet-in-the-middle attack. To solve the problem, [2] chooses additional neutral words from a chaining variable. Anyway, the condition for the neutral words is very restrictive for the local collision technique.

A variant of the local collision was introduced in [12] and generalized to the *initial structure* [11]. As opposed to [2], [11] introduced the *efficient consistency check* technique for the initial structure and it can also be used for the local collision technique. In regard to the technique in [2], the consistency for local collisions is satisfied randomly after matching the meet-in-the-middle attack, while the efficient consistency check satisfies the consistency for the initial structure at the same time as the meet-in-the-middle attack by adding a word for the table used by the meet-in-the-middle-attack.

Similar to partial-fixing, we can ignore d steps regarding neutral words for the starting part in the meet-in-the-middle attack. How to construct an initial structure is still somewhat ambiguous. With several manual attempts, it seems possible to construct d-step initial structures up to ≈ 4 for the case of SHA-b. An example of the initial structure is provided in a later section.

Summary. Considering the meet-in-the-middle attack, we can use all of the techniques described above: splice-and-cut, partial-matching and -fixing, and initial structure. Figure 1 shows how to partition the steps in SHA-b with these techniques in an abstract model.

3 Analysis of Linear Message Schedule

The message schedule of SHA-b is different from that for MD4 and MD5, which were already attacked [5,11], and is essentially linear for w_j $(j \geq 16)$ from Eq.(1). Similarly, HAS-160 adopts a linear message schedule, but most part of the message schedule is only permutations of message words. In fact, only one fifth of

w_js are essentially linear, and this linear w_js are only XOR of 4 message words. Thus, for example, the case of $w_{16} = m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$ is regarded such that all of m_{12}, m_{13}, m_{14}, m_{15} are used in this step in the attack [12]. While, on the message schedule of SHA-b, w_j $(0 \leq j < 16)$ is equal to m_j and seems simple, but w_j $(j \geq 20)$ depends on almost half the number of m_js since w_j $(j \geq 16)$ is computed using Eq.(1). So, it seems that we can compute a preimage up to ≈ 39 steps $(= 20 + 15 + 4)$ faster than the brute-force attack under the same strategy in [12], and it seems difficult to increase the number of steps that can be attacked. This section presents a way to address this problem, that is, the following section finds the chunks that satisfy Condition (3) and detect the neutral words in the chunks.

3.1 Kernel and Neutral Words

This section describes how to partition steps into chunks and find neutral words for SHA-0. For SHA-1, the same approach can be applied by considering bits instead of words.

The expanded message, w_j, is computed using Eq.(1), and its matrix representation is given hereafter: $[w_0\ w_1\ \cdots\ w_{79}]^T = WM^T$, where $M = [m_0\ m_1\ \cdots\ m_{15}]$ and W is represented in Figure 3. Consider that SHA-0 is reduced to s steps and the steps are partitioned into the following two chunks.

$$\begin{cases} [w_0\ w_1\ \cdots\ w_{t-1}]^T = W_1 M^T \\ [w_t\ w_{t+1}\ \cdots\ w_{s-1}]^T = W_2 M^T \end{cases} \tag{4}$$

We assume that

$$\begin{cases} \operatorname{rank} W_1 < 16 \\ \operatorname{rank} W_2 < 16 \end{cases} \tag{5}$$

holds. So, there exists the following non-trivial kernels.

$$\begin{cases} \ker W_1 = \langle k_1^{(0)}, k_1^{(1)}, \ldots, k_1^{(\kappa_1 - 1)} \rangle \\ \ker W_2 = \langle k_2^{(0)}, k_2^{(1)}, \ldots, k_2^{(\kappa_2 - 1)} \rangle \end{cases}, \tag{6}$$

where κ_1 and κ_2 denote the dimension of the corresponding kernel. Let $K_1 = [k_1^{(0)}\ k_1^{(1)}\ \cdots\ k_1^{(\kappa_1-1)}]$ and $K_2 = [k_2^{(0)}\ k_2^{(1)}\ \cdots\ k_2^{(\kappa_2-1)}]$. We regard the message words corresponding to the vectors in K_1 and K_2 as neutral words for the opposite chunk. Consider the following as an example. $\kappa_1 = \kappa_2 = 1$ and

$$\begin{cases} k_1 = [1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T \\ k_2 = [0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T \end{cases}.$$

Since k_1 and k_2 are in the kernel of W_1 and W_2, $W_1 k_1 = 0$ and $W_2 k_2 = 0$ holds. That is, m_0 can be used as a neutral word for the second chunk with $m_0 = m_2 = m_3$ to see '1' in k_1, and m_1 can be used as a neutral word for the first chunk with $m_1 = m_4$ to see '1' in k_2. Similarly, we can choose neutral words whenever the representation of the generating vectors does not share the same position of '1's. However, the strategy does not always work in a straightforward manner. We notice the case that the generating vectors share the '1' at the same position. In this case, we cannot independently determine the value of neutral words for each chunk. We can solve this problem using a sophisticated linear transformation by substituting M with M', where $M^T = RM'^T$ with regular matrix R. Once we find M', we can easy to recover the preimage M by multiplying the matrix R.

Let the unit vector be $\mathbf{e}_i = [0 \ \cdots \ \overset{i}{\check{1}} \ \cdots \ 0]^T$ and j-dimensional identity matrix be E_j. In the following, we construct regular matrix R such that

$$\begin{cases} W_1 R\mathbf{e}_i &= 0 \quad \text{for } i = 0, 1, \ldots, \kappa_1 - 1 \\ W_2 R\mathbf{e}_{i+\kappa_1} &= 0 \quad \text{for } i = 0, 1, \ldots, \kappa_2 - 1 \end{cases}. \tag{7}$$

If such a matrix is constructed, we have

$$[w_0 \ w_1 \ \cdots \ w_{s-1}]^T = \left[\frac{W_1}{W_2}\right] M^T = \left(\left[\frac{W_1}{W_2}\right] R\right)(R^{-1}M^T).$$

Let $W_1' = W_1R$, $W_2' = W_2R$, $M'^T = R^{-1}M^T$, and $M' = [m_0' \ m_1' \ \cdots \ m_{15}']$, and we have

- $\ker W_1' = \langle \mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{\kappa_1 - 1}\rangle$, and $\ker W_2' = \langle \mathbf{e}_{\kappa_1}, \mathbf{e}_{\kappa_1+1}, \ldots, \mathbf{e}_{\kappa_1+\kappa_2-1}\rangle$.
- $m_0', m_1', \ldots, m_{\kappa_1-1}'$ are neutral words for the second chunk, and $m_{\kappa_1}', m_{\kappa_1+1}', \ldots, m_{\kappa_1+\kappa_2-1}'$ are neutral words for the first chunk.

Thus, we can perform the meet-in-the-middle attack described in Section 2.3 by adjusting a recovered preimage M' with $M^T \leftarrow RM'^T$. The rest of this section describes how to construct R.

Assume $\operatorname{rank}[K_1 \ K_2] = \kappa_1 + \kappa_2$[2]. We can choose $\kappa_1 + \kappa_2$ independent row vectors in $[K_1 \ K_2]$, and there is regular matrix T that collects these independent row vectors and can be constructed from E_{16} by swapping corresponding rows, H, at the top by swapping rows, and regular matrices B and S are defied as follows.

$$\left[\begin{array}{c} H \\ \hline \\ * \\ \\ \end{array}\right] = T[K_1 \ K_2], \ B = \left[\begin{array}{c|c} H^{-1} & 0 \\ \hline 0 & E_{16-\kappa_1-\kappa_2} \end{array}\right], \ S = \left[\begin{array}{c|c} BT[K_1 \ K_2] & \begin{array}{c} 0 \\ \hline E_{16-\kappa_1-\kappa_2} \end{array} \end{array}\right].$$

Note that the top $\kappa_1 + \kappa_2$ rows of $BT[K_1 \ K_2]$ is $E_{\kappa_1+\kappa_2}$. Then, $R = T^{-1}B^{-1}S$ satisfies $k_1^{(i)} = R\mathbf{e}_i$ (for $0 \le i < \kappa_1$) and $k_2^{(i)} = R\mathbf{e}_{i+\kappa_1}$ (for $0 \le i < \kappa_2$). So, Eq.(7) holds.

[2] Of course, there is a chunk partition such that $\operatorname{rank}[K_1 \ K_2] < \kappa_1 + \kappa_2$; however, we are not so interested in this case. Actually, we do not have an experience with $\operatorname{rank}[K_1 \ K_2] < \kappa_1 + \kappa_2$ with long steps.

Table 2. Number of Steps Such That rank W_1, rank $W_2 < 16$ for SHA-0

$c+d$	0	1–2	3	4–6	7	8–11	12–13	14–15	16–21	22	23	24–25	26–27	28
# of steps	32	33	35	37	39	42	45	47	52	54	55	57	60	61

c: number of partial-fixing step, d : number of initial structure step.

Table 3. Number of Steps Such That rank W_1, rank $W_2 < 512$ for SHA-1

$c+d$	0	1–2	3	4–6	7	8–11	12–13	14–15	16–21	22	23	24–25	26–27	28
# of steps	32	33	35	37	39	42	45	47	52	54	55	57	60	61

c: number of partial-fixing step, d : number of initial structure step.

3.2 Notes on Auxiliary Techniques

Both the splice-and-cut and partial-matching techniques described in Section 2.4 can be used in the same way. Note, we generate pseudo-preimages in the same way as Section 2.4, because the splice-and-cut technique cannot specify IV.

We can apply the partial-fixing and initial structure techniques described in Section 2.4 to SHA-b in a similar way. However, careful analysis is required, since the message schedule of SHA-b sometimes produces XOR of several message words in one step.

[12] applies the partial-fixing technique to HAS-160. The step function of HAS-160 is very similar to SHA-b, so these techniques can also be applied to SHA-b.

3.3 Application to SHA-*b*

Based on the discussion above, we compute how many steps to satisfy Condition (5), with partial-matching and -fixing step $c \leq 21$ and initial structure step $d \leq 7$. The results are shown in Tables 2, 3, and 4. We do not know why, but we notice that the numbers of steps are the same when the values of $c + d$ are the same.

For SHA-1, rank W_1, rank $W_2 < 512$ is a very hard condition to attack SHA-1, because we may be able to use only one neutral bit. In this case the partial-fixing technique cannot work. So, we also compute the case that rank W_1, rank $W_2 < 503$ to have the possibility to use the partial-fixing technique. Though we loose the upper bound of the rank to 503, the derived ranks are 480.

Consider the case for SHA-0. If the number of steps in chunks are 15, rank W_1, rank $W_2 < 16$ always holds. To set $d = 0$ and $c = 4$, that is, we do not use initial

Table 4. Number of Steps Such That rank W_1, rank $W_2 < 503$ for SHA-1

$c+d$	0–1	2	3–5	6	7	8–9	10–11	12–13	14–15	16–17	18	19	20–21	22–24	25	26–27	28
# of steps	31	33	35	36	37	39	41	43	45	47	48	49	51	54	56	57	59

c: number of partial-fixing step, d : number of initial structure step.

structure and partial-fixing, and the attack always works. Thus, we can trivially compute a preimage of the compression function reduced to 34 ($= 15 + 15 + 4$) steps in 2^{128}. To see Table 2, we see 37 when $c + d = 4$. So, we can improve the attack to 37 steps with the same complexity. Consider to adopt the partial-fixing technique. To fix lower 16 bits in neutral words, it is easy to verify that we can increase 3 more steps. Following Table 2 with $c + d = 7$, we can compute a preimage of the compression function reduced to 39 steps in 2^{144}.

Note that Condition (5) is the only necessary condition for a successful attack. To construct a definite attack procedure, we need to see specific procedures for the initial structure, and for partial-fixing, and for padding. The following section describes this.

4 Detailed Attack Against SHA-0

This section describes detailed description of the attack against SHA-0 reduced to 52 steps. We try to increase the number of steps that can be attacked faster than the brute-force attack as large as possible. For smaller number of steps, see the previous section.

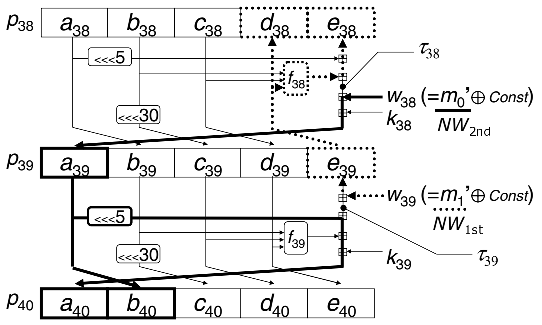
4.1 Chunk Partition for 52-Step SHA-0

The transformed message schedule, WR, described in the previous section is shown in Table 5. As shown in Table 5, the first chunk (steps 37–23, in total 15 steps) includes m'_1 but does not include m'_0, and the second chunk (steps 40–51,

Table 5. Transformed Message Schedule for 52-step SHA-0

	m'_i																
Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Second chunk
1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
5	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
8	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
9	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Skip
10	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
11	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
12	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
13	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
16	1	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	
17	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	
18	1	1	1	0	1	0	0	0	0	0	1	0	0	0	0	1	
19	0	0	1	1	0	1	0	0	1	0	0	1	0	1	0	0	
20	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	0	
21	0	0	1	0	1	1	0	1	0	0	1	0	0	1	0	1	
22	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1	0	
23	0	1	0	1	1	0	1	1	0	1	0	0	1	0	1	1	First chunk
24	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	
25	0	1	1	0	0	1	1	0	0	1	0	0	0	1	0	0	
26	0	0	1	1	0	0	1	1	0	1	0	0	0	0	1	0	
27	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	1	
28	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	
29	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	
30	0	1	1	0	1	1	1	1	0	0	0	1	0	0	0	1	
31	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	
32	0	1	0	1	1	0	1	1	1	1	0	0	0	1	1	0	
33	0	1	1	0	1	1	0	1	1	0	0	0	0	0	1	1	
34	0	1	1	1	0	1	1	0	0	0	1	0	0	1	0	1	
35	0	1	1	1	1	0	1	1	1	1	1	1	0	1	1	0	
36	0	1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	
37	0	1	0	1	1	1	1	0	0	0	1	0	1	0	0	1	↑
38	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	IS
39	0	1	1	1	0	1	1	1	1	1	0	1	1	0	0	0	
40	0	0	1	1	1	0	1	1	1	0	1	0	1	1	0	0	↓ Second chunk
41	1	0	0	1	1	1	0	1	1	0	1	1	0	1	1	0	
42	0	0	0	0	1	1	1	0	1	0	1	1	1	0	1	1	
43	0	0	1	0	0	1	1	1	1	0	1	1	1	0	0	1	
44	1	0	1	1	0	0	1	1	0	1	1	1	1	0	0	0	
45	0	0	1	1	1	0	0	1	1	1	1	1	1	1	0	0	
46	1	0	1	1	1	1	0	0	1	0	0	1	1	1	1	0	
47	1	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	
48	0	0	1	0	1	1	1	1	1	0	1	1	0	0	1	1	
49	0	0	1	1	0	1	1	1	0	1	0	1	1	1	0	1	
50	1	0	0	1	1	0	1	1	0	1	1	0	1	0	1	0	
51	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	1	

w_i consists of XOR of m'_j whose entry in step i is 1, e.g., $w_{10} = m'_1 \oplus m'_{10}$. IS, which appears in steps 38 and 39, stands for "Initial Structure."



The bold and dotted lines represent data lines for which values are changed depending on the value of the neutral words for the second and first chunks, respectively. Narrow lines represent data lines that are always fixed regardless of the values of the neutral words.

Fig. 2. Initial structure for 52-step SHA-0

0–8 in total 21 steps) includes m'_0 but does not include m'_1. Hence, by fixing m'_2 to m'_{15}, the meet-in-the-middle attack can be performed.

4.2 Initial Structure for 52-Step SHA-0

The construction of the initial structure is shown in Fig. 2. The goal of this construction is making p_{40} independent of the neutral words for the first chunk w_{39} $(= m'_1 \oplus \mathit{Const})$, and making p_{38} independent of the neutral words for the second chunk w_{38} $(= m'_0 \oplus \mathit{Const})$. This is achieved by the following procedure.

Preparation: Change the addition order in step 39, and choose an arbitrary value for τ_{38} and τ_{39}, e.g. $\tau_{38} = \tau_{39} = 0$. Moreover, fix a_{38}, b_{38}, c_{38} as arbitrary.

Make p_{40} independent of w_{39}: c_{40} $(= a_{38}{}^{\lll 30})$, d_{40} $(= b_{38}{}^{\lll 30})$, and e_{40} $(= c_{38})$ are already fixed. Compute b_{40} $(= a_{39} = \tau_{38} + w_{38} + k_{38})$ and a_{40} $(= \tau_{39} + a_{39}{}^{\lll 5} + f_{39}(b_{39}, c_{39}, d_{39}) + k_{39})$. Note that $b_{39} = a_{38}$, $c_{39} = b_{38}{}^{\lll 30}$, and $d_{39} = c_{38}$.

Make p_{38} independent of w_{38}: Compute d_{38} $(= \tau_{39} - w_{39})$ and e_{38} $(= \tau_{38} - f_{38}(b_{38}, c_{38}, d_{38}) - a_{38}{}^{\lll 5})$.

As described above, we can compute the first and second chunks independently of the neutral words for the second and first chunks, respectively. Hence, the meet-in-the-middle attack can be performed.

Remarks. Construction of the initial structure is dependent on the selected chunks. Since the chunk partition is different for SHA-1, we construct the initial structure of SHA-1 differently. See Section 5 for details.

4.3 Partial-Fixing Technique for 52-Step SHA-0

In the meet-in-the-middle attack, results of two chunks must be compared efficiently. Although many steps (14 steps) between two chunks are skipped in the

Table 6. Number of Known Bits in Partial-Fixing Technique for 52-Step SHA-0

j	a_j	b_j	c_j	d_j	e_j	(in w_j) m'_1	#cands of a_j	
9	All	All	All	All	All	18–0		Forward
10	18–0	All	All	All	All	18–0	1	
11	18–5	18–0	All	All	All	18–0	2^1	
12	$\underline{\text{18–10}}$	$\underline{\text{18–5}}$	16–0	All	All	18–0	2^2	
12	$\underline{\text{20–9}}$	$\underline{\text{20–11}}$	?	?	?	skipped		Backward computation
13	20–9	20–9	18–9	?	?	skipped		
14	20–9	20–9	18–7	18–9	?	skipped		
15	20–4	20–9	18–7	18–7	18–9	All	2^9	
16	20–2	20–4	18–7	18–7	18–7	18–0	2^7	
17	20–2	20–2	18–2	18–7	18–7	All	2^5	
18	20–2	20–2	18–0	18–2	18–7	18–0	2^3	
19	All	20–2	18–0	18–0	18–2	All	2^1	
20	All	All	18–0	18–0	18–0	18–0	1	
21	All	All	All	18–0	18–0	All	1	
22	All	All	All	All	18–0	18–0	1	
23	All	All	All	All	All			
j	a_j	b_j	c_j	d_j	e_j	(in w_j) m'_0	#cands of e_j	

Numbers denote the known bits of each chaining variable. Underlined variables in $j = 12$ are variables where we compare the results of two chunks.

Table 7. Number of Known Bits in Partial-Fixing Technique for 48-Step SHA-1

j	a_j	b_j	c_j	d_j	e_j	(in w_j) m'_1	#cands of a_j	
9	All	All	All	All	All	17–0		Forward
10	17–0	All	All	All	All	All	1	
11	17–5	17–0	All	All	All	All	2^1	
12	$\underline{\text{17–10}}$	$\underline{\text{17–5}}$	15–0	All	All	16–0	2^2	
12	$\underline{\text{27–9}}$	$\underline{\text{27–11}}$	?	?	?	skipped		Backward computation
13	27–7	27–9	25–9	?	?	skipped		
14	27–9	27–7	25–7	25–9	?	skipped		
15	27–4	27–9	25–7	25–7	25–9	All	2^9	
16	27–2	27–4	25–7	25–7	25–7	25–0	2^7	
17	27–2	27–2	25–2	25–7	25–7	All	2^5	
18	27–2	27–2	25–0	25–2	25–7	25–0	2^3	
19	All	27–2	25–0	25–0	25–2	All	2^1	
20	All	All	25–0	25–0	25–0	25–0	1	
21	All	All	All	25–0	25–0	All	1	
22	All	All	All	All	25–0	25–0	1	
23	All	All	All	All	All			
j	a_j	b_j	c_j	d_j	e_j	(in w_j) m'_0	#cands of e_j	

We compare results of two chunks on a_{12} and b_{12}, in total 15 bits.

employed attack as shown in Table 5, a part of the results of two chunks can be compared by using the partial-fixing and partial-matching techniques. How the results of two chunks are compared is explained in Table 6. Note we first assumed that the fixed bit-positions for backward computation is represented by lower x bits and the forward computation is represented by intermediate y bits. Then, we identified the best x, y, and fixed positions. Consequently, we chose $x = 19$ and $y = 19$ from the least significant bit.

We explain how the partial computation shown in Table 6 is processed.

Forward computation for a_{10}: As a result of computing the second chunk in forward direction m'_0, we obtain the value p_9. Therefore, when we apply partial-fixing to the forward computation, we know all bits of a_9, b_9, c_9, d_9 and e_9. p_{10} is computed with $R_9(p_9, w_9)$, where w_9 can be written as $m'_1 \oplus$ *Const*. Since the lower 19 bits of m'_1, which is the neutral word for the other chunk, are fixed, the lower 19 bits of a_{10} can be computed uniquely.

Forward computation for a_{11}: p_{11} is computed with $R_{10}(p_{10}, w_{10})$, where w_{10} can be written as $m'_1 \oplus$ *Const*. In particular, the equation for a_{11} is as follows:

$$a_{11} = a_{10}{}^{\lll 5} + f_{10}(b_{10}, c_{10}, d_{10}) + e_{10} + w_{10} + k_{10}.$$

Since the lower 19 bits of w_{10} and all bits of f_{10}, e_{10}, and k_{10} are known, the lower 19 bits of $f_{10} + e_{10} + w_{10} + k_{10}$ can be computed uniquely. We know the lower 19 bits (bits 0 to 18) of a_{10}, hence we know bits 5 to 23 of $a_{10}{}^{\lll 5}$. When we compute $a_{11} = a_{10}{}^{\lll 5} + (f_{10} + e_{10} + w_{10} + k_{10})$, we do not know if there is a carry from bit-position 4 to 5. Therefore, we consider both possible carry bits, and obtain two candidates for bits 5 to 18 of a_{11}. Hence, for each $(a_9, b_9, c_9, d_9, e_9)$, we obtain 2^1 candidates for bits 5 to 18 of a_{11}.

Forward computation for a_{12}: By almost the same procedure as above, we can obtain two candidates for bits 10 to 18 a_{12} for each candidate of p_{11}. Hence, for each $(a_9, b_9, c_9, d_9, e_9)$, we obtain 2^2 candidates for bits 10 to 18 of a_{12}.

Backward computation for e_{22}: As a result of computing the first chunk in backward direction m'_1, we obtain the value of p_{23}. p_{22} is computed with $R_{22}^{-1}(p_{23}, w_{22})$, where w_{22} can be written as $m'_0 \oplus \mathit{Const}$. Since the lower 19 bits of m'_0 are fixed, the lower 19 bits of e_{22} can be computed uniquely.

Backward computation for e_{17}: With similar techniques to the forward computation, we can compute 2^3 candidates for p_{18} as shown in Table 6 for each p_{23}. We next explain how to compute p_{17} with $R_{17}^{-1}(p_{18}, w_{17})$, in particular,

$$e_{17} = a_{18} - k_{17} - w_{17} - f_{17}(c_{18}{}^{\ggg 30}, d_{18}, e_{18}) - b_{18}{}^{\lll 5},$$
$$w_{17} = m'_1 \oplus \mathit{Const} = m'_1 \oplus m'_3 \oplus m'_9 \oplus m'_{14}.$$

In order to reduce the number of unknown carries, the number of additions (or subtractions) should be reduced as much as possible. For this purpose, we fix the lower 19 bits of w_{17} to $-k_{17}$. This can be achieved by first fixing the lower 19 bits of $m'_1 \oplus m'_3 \oplus m'_9$, and then compute $m'_{14} = m'_1 \oplus m'_3 \oplus m'_9 \oplus (-k_{17})$ with respect to the lower 19 bits.

Remarks for the rest: In a similar manner, we obtain Table 6. Note, we need to fix $w_{16} = m'_0 \oplus m'_1 \oplus m'_2 \oplus m'_8 \oplus m'_{13}$ to $-k_{16}$ and $w_{15} = m'_{15}$ to $-k_{15}$ with respect to the lower 19 bits. With adequate message space, this can be easily achieved. Backward computation is done until p_{15}. Steps 14-11 are skipped in the partial-matching technique. Finally, we compare the results from both chunks at bits 10–18 of a_{12} and bits 11–18 of b_{12}, in total 17 bits.

4.4 Attack Procedure for 52-Step SHA-0

For a given hash value, H_m, the attack procedure is as follows.

1. Fix $m'_i, (i \notin \{0, 1\})$ and the lower 19 bits of m'_0 and m'_1 to randomly chosen values.
2. Fix chaining variables in the initial structures (steps 38-39) as described in Section 4.2.
3. For all 13 free bits of the neutral words for the second chunk, namely the higher 13 bits of m'_0,
   (a) Compute a_{40} and b_{40} from w_{38} as explained in Section 4.2.
   (b) Compute: $\begin{cases} p_{j+1} \leftarrow R_j(p_j, w_j) & \text{for } j = 40, 41, \ldots, 51 \\ p_0 \leftarrow H_m - p_{52}, & \\ p_{j+1} \leftarrow R_j(p_j, w_j) & \text{for } j = 0, 1, \ldots, 8 \end{cases}$
   (c) Compute bits 0–18 of a_{10}, 2^1 candidates for bits 5–18 of a_{11} and 2^2 candidates for bits 10–18 of a_{12} as explained in Section 4.3.

   (d) Make a table of $(m'_0, p_9, a_{10}, a_{11}, a_{12})$s. Since we have 13 free bits in neutral words, and 2^2 candidates of partial a_{12} for each choice of free bits, we have 2^{15} items in the table.
4. For all 13 free bits of the neutral words for the first chunk, namely the higher 13 bits of m'_1,
   (a) Compute e_{38} and d_{38} as described in Section 4.2.
   (b) Compute: $p_j \leftarrow R_j^{-1}(p_{j+1}, w_j) \qquad \text{for } j = 37, 36, \ldots, 23,$
   (c) Compute the lower 19 bits of e_{22}, e_{21}, and e_{20}, bits 2–18 of e_{19}, bits 7–18 of e_{18}, e_{17}, and e_{16}, and bits 9–18 of e_{15} as explained in Section 4.3.
   (d) For each item in the table, check whether or not bits 10–18 of a_{12}, and bits 11–18 of b_{12} computed from both chunks match.
   (e) If a match is found, compute p_{10} to p_{13} by the corresponding message word, and check the match of the additionally computed bits, and check the correctness of the guess for the carry for a_{11} and a_{12} step by step.
   (f) If a match is found, compute p_{22} to p_{11} by the corresponding message word, and check whether all values from both chunks match and check the correctness of the guess for the carry for e_{19} to e_{15}.
   (g) If all bits match, the pair of the corresponding message and p_0 is a pseudo-preimage.

4.5 Complexity Estimation for 52-Step SHA-0

Assume the complexity for computing 1 step is $\frac{1}{52}$ 52-step SHA-0 compression function, and the memory access cost is negligible compared with the cost of the computation of the step function.

– The complexity of Steps 1 and 2 are negligible.
– The complexity of Step 3a is approximately $2^{13} \cdot \frac{2}{52}$.
– The complexity of Step 3b is approximately $2^{13} \cdot \frac{21}{52}(= 2^{13} \cdot \frac{12}{52} + 2^{13} \cdot \frac{9}{52})$.
– The complexity of Step 3c is approximately $2^{13} \cdot \frac{7}{52}$ $(=2^{13}(\frac{1}{52} + 2^1 \cdot \frac{1}{52} + 2^2 \cdot \frac{1}{52}))$.
– The complexity of Step 4a is approximately $2^{13} \cdot \frac{2}{52}$.
– The complexity of Step 4b is $2^{13} \cdot \frac{15}{52}$.
– The complexity of Step 4c is approximately $2^{13} \cdot \frac{685}{52}$ $(=2^{13}(\frac{1}{52} + \frac{1}{52} + \frac{1}{52} + 2^1 \cdot \frac{1}{52} + 2^3 \cdot \frac{1}{52} + 2^5 \cdot \frac{1}{52} + 2^7 \cdot \frac{1}{52} + 2^9 \cdot \frac{1}{52}))$.
– The first chunk produces $2^{22}(= 2^{13} \cdot 2^9)$ items. Therefore, at Step 4d, $2^{37}(= 2^{22} \cdot 2^{15})$ pairs are compared and 2^{20} $(=2^{37} \cdot 2^{-17})$ pairs will remain.
– At Step 4e, the complexity of computing p_{10} and p_{11} is approximately $2^{20} \cdot \frac{2}{52}$. Then, by comparing two additional bits of a_{11} (bit-positions 19 and 20) and checking the correctness of the 1 guess for the carry for a_{11}, the number of remaining pairs becomes 2^{17} $(=2^{20} \cdot 2^{-3})$. The complexity of computing p_{12} is approximately $2^{17} \cdot \frac{1}{52}$ and by comparing three additional bits of a_{12} (bit-positions 9, 19, and 20) and checking the correctness of the 1 guess of carry for a_{12}, the number of remaining pairs becomes 2^{13} $(=2^{17} \cdot 2^{-4})$. The complexity of computing p_{13} is approximately $2^{13} \cdot \frac{1}{52}$ and by comparing twelve additional bits of a_{13} (bit-positions 9-20), the number of remaining pairs becomes 2^1 $(=2^{13} \cdot 2^{-12})$.

$$W = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
1&0&1&0&0&0&0&0&1&0&0&0&0&1&0&0\\
&&&&&&&&\vdots&&&&&&&
\end{bmatrix}$$

Fig. 3. SHA-0 Message Schedule in Matrix Form

$$R = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}$$

Fig. 4. Linear Transformation Used in the Attack of Reduced SHA-0

– Complexity for Step 4f is negligible since the number of remaining pairs is sufficiently reduced compared to the previous part. By checking the correctness of the guesses of carry for e_{22} to e_{15}, the number of remaining pair becomes 2^{-8} $(=2^1 \cdot 2^{-9})$. By checking the entire p_{13} for matches, the number of remaining pair becomes 2^{-134} $(=2^{-8} \cdot 2^{-126})$. Therefore, by repeating this attack 2^{134} times, we can expect to find a pseudo-preimage.

The complexity for Step 3 is $2^{13}(\frac{2}{52} + \frac{21}{52} + \frac{7}{52}) = 2^{13} \cdot \frac{30}{52}$. The complexity for Step 4 is $2^{13}(\frac{2}{52} + \frac{15}{52} + \frac{685}{52}) + 2^{20} \cdot \frac{2}{52} + 2^{17} \cdot \frac{1}{52} + 2^{13} \cdot \frac{1}{52} = 2^{13}(\frac{702}{52} + \frac{256}{52} + \frac{16}{52} + \frac{1}{52}) = 2^{13} \cdot \frac{975}{52}$. Hence, we can find a pseudo-preimage with a complexity of $2^{13}(\frac{30}{52} + \frac{975}{52}) \cdot 2^{134} = 2^{17.20} \cdot 2^{134} \approx 2^{151.2}$. This can be converted to the preimage attack with a complexity of $2^{156.6}$ by using the algorithm described in Section 2.2.

In this attack, we use a memory to store 2^{15} $(m'_0, p_9, a_{10}, a_{11}, a_{12})$s in Step 3d. Therefore, the memory complexity of this attack is approximately $2^{15} \times 9$ words. To apply the technique [6, Remark 9.93], we can convert our attack into memoryless version. The converted attack requires negligible memory and finds a pseudo-preimage in $2^{152.2}$. The attack is converted to the preimage attack by using the algorithm described in Section 2.2 with complexity of $2^{157.1}$ with $5 \times 2^{3.9}$ words of memory, which is negligible.

4.6 Padding Issue

In the attack described above, we use R in Figure 4. Focusing on the padding part with $M^T = RM'^T$, we should satisfy the padding rule with

$$m_{13} = m'_1 \oplus m'_{13}, \qquad m_{14} = m'_{14}, \qquad m_{15} = m'_{15}. \tag{8}$$

- $m'_{14} \leftarrow 0$. When m'_{14} is used in the partial-fixing technique, m'_{14} is XORed with other m'_js. So, there is room to fix m'_{14}. This means the number of message strings is less than 2^{32} bits.
- Set the least significant bit of m_{13} $(= m'_1 \oplus m'_{13})$ to '1'. Although m'_1 is a neutral word, the partial-fixing technique fixes the least significant 20 bits. By appropriately setting the least significant bit of m'_{13}, this condition is satisfied.
- $m'_{15} \bmod 2^9 \leftarrow 447$. This agrees with the padding rule for m_{13}. However, we specified $m'_{15} = -k_{15}$ in the attack procedure when we perform the partial-fixing technique for step 15. To observe $m'_{15} + k_{15}$, the least significant three bits are zero. Additionally, since we know 21-2 bits of a_{16}, we can determine the carry from the 8th bit to the 9th bit. This is the same effect as setting $m'_{15} = -k_{15}$.

In conclusion, we can compute a pseudo-preimage following the padding rule at the same complexity, $2^{151.2}$ as describe above, and we can compute a 2-block preimage with the regular padding in $2^{156.6}$.

Note, even if the padding rule cannot be satisfied, the attack is valid as a second-preimage attack.

5 Attack Sketch for 48-Step SHA-1

This section describes the sketch of the attack against SHA-1 reduced to 48 steps.

5.1 Chunk Partition

Let E be E_{32}. The transformed message schedule, $W' = WR$, is shown in Table 8. In SHA-1, the size of W' is 512. We searched for chunk partition of 48-step SHA-1, and found the pattern where κ_1 and κ_2 in Eq.(6) are 32. When we attack SHA-1, we use the first 64 bits of M' as neutral words, and fix the other 448 bits. Hence, we show only the first 64 bits of W'.

5.2 Initial Structure and Partial-Fixing Technique

The construction of the initial structure is shown in Fig. 5. To fix the output of f_2, we use the cross absorption property presented by [11]. We manually optimized the number of n in the initial structure shown in Fig. 5 by considering the efficiency of the partial-fixing technique together. As a result, we select $n = 24$.

The partial-fixing technique for 48-step SHA-1 skips 14 steps as shown in Table 8. How the results of two chunks are compared is explained in Table 7. In forward computation, we fix the lower 18 bits of m'_1, and in backward computation, we fix the lower 26 bits of m'_0. Finally, bit positions 10 to 17 of a_{12} and bit positions 11 to 17 of b_{12}, in total 15 bits, are compared.

Table 8. Transformed Message Schedule for 48-step SHA-1

Step	1st 32 cols of W'	2nd 32 cols of W'	
0	$E \oplus E^{\lll 2}$	0	IS
1	0	E	
2	E	0	
3	0	E	
4	$E^{\lll 1}$	0	↓ Second chunk
5	0	0	
6	E	0	
7	E	0	
8	$E^{\lll 1}$	0	
9	0	0	
10	E	0	
11	E	0	
12	$E^{\lll 1}$	0	
13	0	0	
14	0	0	
15	E	0	
16	$E \oplus E^{\lll 1}$	0	
17	0	0	
18	E	0	
19	E	$E^{\ggg 1}$	Skip
20	0	0	
21	0	0	
22	E	$E^{\ggg 2}$	
23	0	0	
24	0	0	
25	0	$E^{\ggg 3}$	
26	E	0	
27	0	$E^{\ggg 2}$	
28	E	$E^{\ggg 4}$	
29	0	0	
30	E	0	
31	0	$E^{\ggg 5}$	
32	E	0	
33	0	$E^{\ggg 2} \oplus E^{\ggg 4}$	First chunk
34	0	$E^{\ggg 6}$	
35	0	$E^{\ggg 2} \oplus E^{\ggg 3}$	
36	0	0	
37	0	$E^{\ggg 7}$	
38	0	$E^{\ggg 4}$	
39	0	$E^{\ggg 4} \oplus E^{\ggg 6}$	
40	0	$E^{\ggg 8}$	
41	0	$E^{\ggg 4}$	
42	0	0	
43	0	$E^{\ggg 4} \oplus E^{\ggg 9}$	
44	0	0	
45	0	$E^{\ggg 6} \oplus E^{\ggg 8}$	
46	0	$E^{\ggg 10}$	
47	0	$E^{\ggg 3} \oplus E^{\ggg 6} \oplus E^{\ggg 11}$	↑

The second and the third columns of the table show the first and second 32 columns of W'. In each step, 32 rows of W' are related. Hence, each entry of the table denotes corresponding 32×32 submatrix of W'.

Since all 'j's of E_j used in this table are 32, we simply write E to denote E_{32}.

5.3 Summary of Attack

In this attack, a_4 and m'_0 are the neutral words for the second chunk where, m'_0 is the first 32 bits of M'. Similarly, b_0 and m'_1 are the neutral words for the first chunk, where, m'_1 is the second 32 bits of M'.

To construct the initial structure appropriately and apply the partial-fixing technique efficiently, we need to fix a part of neutral words. In the first chunk, we fix bit positions 26, 27, 28, 29, 30, 31, 0, and 1, in total 8 bits, of b_0s. This results in fixing the upper 8 bits of c_1, which is necessary for the initial structure. We also fix bit positions 1 to 18 of, in total 18 bits, of m'_1. This results in fixing the lower 18 bits of w_{19}, which is a message word used in the first step in the partial-fixing technique in forward direction. Note, the number of unfixed bits in neutral words for the first chunk is 38. In the second chunk, we fix the lower 26 bits of m'_0. This result in fixing the lower 24 bits of w_0 $(= (E \oplus E^{\lll 2}) \times m'_0)$, which is required for the initial structure, and fixing the lower 24 bits of w_j $(= E), j \in \{32, 30, 28, 26\}$, which is required for the partial-fixing technique.

We roughly estimate the complexity of the attack. Considering the unknown carries in the partial-fixing, the meet-in-the-middle attack examines the match of 2^{87} $(= 2^{38+2} \times 2^{38+9})$ pairs. Unfortunately, since we can only match 47 bits, the number of resulting pairs are $2^{40} \gg 2^{38}$. So, the attack requires more time than the brute-force attack. To reduce the time complexity, we analyze the

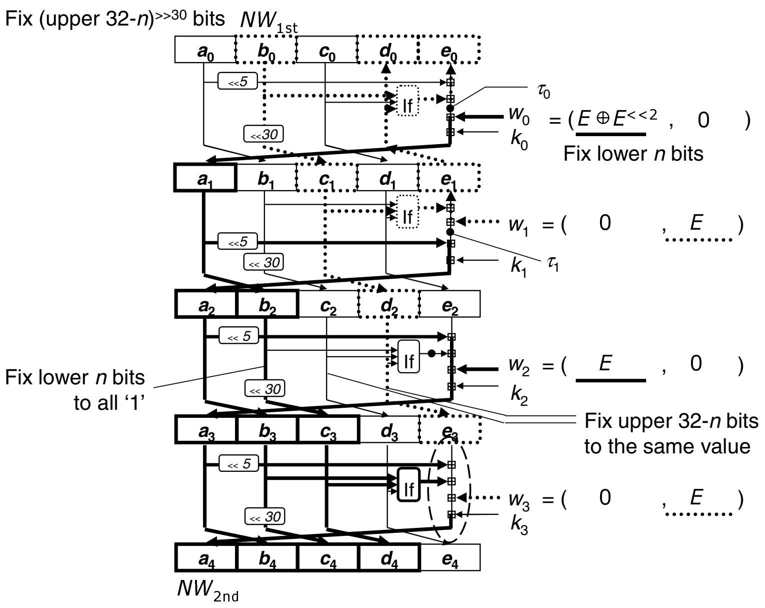


Fig. 5. Initial structure for 48-step SHA-1

Note that we perform the efficient consistency check described in the initial structure part of Section 2.4 in dashed circle in the figure.

probabilistic behavior of carry propagation in the partial-fixing. Observing Table 7, we notice that we can estimate the existence of carry with a probability higher than 1/2 for several additions with unknown carry. In the backward computation, we can estimate the existence of carry with a probability higher than 3/4 for two cases. Thus, the meet-in-the-middle attack examines the match of 2^{85} $(= 2^{38+2} \times 2^{38+7})$ pairs. Since the matching bit is 47 bits, the number of resulting pairs are $2^{38} \approx 2^{38}$. So, the time complexity for the dominant part is computing the chunks, and is approximately 2^{39} and the success probability is approximately $2^{-117.7}$ $(= 2^{32+6+6-160} \times (3/4)^2)$. To iterate the above procedure $2^{117.7}$ times, we find a pseudo-preimage with high probability, and the total time complexity is approximately $2^{156.7}$ $(= 2^{39} \times 2^{117.7})$. Consider a second preimage attack whose block is longer than 3, applying the above attack to the second block, and using the conversion described in Section 2.2, and a preimage will be found in $2^{159.3}$.

In this attack, we use a memory to store 2^{40} items. Therefore, the memory complexity is approximately $2^{40} \times 11$ words.

6 Conclusion

This paper proposes a method for analyzing the linear message schedule in SHA-0 and SHA-1 for a preimage attack using the meet-in-the-middle attack. Thanks to recently developed auxiliary techniques such as splice-and-cut, partial-fixing, and initial structure, the results of the application of the proposed method can be used to compute preimages of reduced SHA-0 and SHA-1 up to 52 and 48 steps, respectively, faster than the brute-force attack. The results shows that the meet-in-the-middle attack is also effective for a linear message schedule compared to permutations of the message words. Since SHA-0 and SHA-1 have 80 steps and the attack described herein does not reach the same number of steps, the preimage resistance of SHA-0 and SHA-1 is still sufficient. We should pay attention to the progress of the techniques related to preimage resistance.

Acknowledgments

The authors would like to thank the anonymous referee for pointing out that we can construct initial structure with deterministic way against 52-step SHA-0. It also reduces the memory complexity by a factor of 2^{32}.

References

1. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R., Keliher, L., Sica, F. (eds.) Selected Areas in Cryptography — Workshop Records of 15th Annual International Workshop, SAC 2008, Sackville, New Brunswick, Canada, pp. 82–98 (2008)
2. Aumasson, J.-P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Avanzi, R., Keliher, L., Sica, F. (eds.) Selected Areas in Cryptography — Workshop Records of 15th Annual International Workshop, SAC 2008, Sackville, New Brunswick, Canada, pp. 99–114 (2008) (also appears in IACR Cryptology ePrint Archive: Report 2008/183, `http://eprint.iacr.org/2008/183`)
3. De Cannière, C., Rechberger, C.: Preimages for reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008) (slides on preliminary results presented at ESC 2008 seminar, `http://wiki.uni.lu/esc/`)
4. Hong, D., Chang, D., Sung, J., Lee, S., Hong, S., Lee, J., Moon, D., Chee, S.: New FORK-256 (2007) (IACR Cryptology ePrint Archive: Report 2007/185, `http://eprint.iacr.org/2007/185`)
5. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
6. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
7. Rivest, R.L.: The MD4 message digest algorithm. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991); Also appears in RFC 1320, `http://www.ietf.org/rfc/rfc1320.txt`
8. Rivest, R.L.: Request for Comments 1321: The MD5 Message Digest Algorithm. The Internet Engineering Task Force (1992), `http://www.ietf.org/rfc/rfc1321.txt`
9. Saarinen, M.-J.O.: A meet-in-the-middle collision attack against the new FORK-256. In: Srinathan, K., Pandu Rangan, C., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 10–17. Springer, Heidelberg (2007)

10. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
11. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Cramer, R. (ed.) Advances in Cryptology — EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
12. Sasaki, Y., Aoki, K.: A preimage attack for 52-step HAS-160. In: Lee, P.J., Cheon, J.H. (eds.) Information Security and Cryptology - ICISC 2008, 11th International Conference. LNCS, vol. 5461, pp. 302–317. Springer, Heidelberg (2009)
13. U.S. Department of Commerce, National Institute of Standards and Technology. Secure Hash Standard (SHS) (Federal Information Processing Standards Publication 180-3) (2008), `http://csrc.nist.gov/publications/PubsFIPS.html#FIPS%20186-3`
14. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
15. Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL — one-way hashing algorithm with variable length of output. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 83–104. Springer, Heidelberg (1993)

Private Mutual Authentication and Conditional Oblivious Transfer

Stanisław Jarecki and Xiaomin Liu

University of California, Irvine

Abstract. A bi-directional Private Authentication, or *Unlinkable Secret Handshake*, allows two parties to authenticate each other as certified by given certification authorities (i.e. affiliated with given groups), in a *mutually private way*, in the sense that the protocol leaks no information about either participant to a party which does not satisfy that participant's authentication policy. In particular, the protocol hides what group this participant belongs to, and protocol instances involving the same participant are unlinkable. We construct the first realization of such private authentication using $O(1)$ exponentiations and bilinear maps, secure under Strong Diffie-Hellman and Decisional Linear assumptions.

Our protocols rely on a novel technical tool, a family of efficient Private Conditional Oblivious Transfer (COT) protocols, secure under DDH, for languages defined by modular arithmetic constraints (e.g. equality, inequality, sums, products) on discrete-log representations of some group elements. (Recall that $(w_1, ..., w_n)$ is a representation of C in bases $(g_1, ..., g_n)$ if $C = g_1^{w_1}...g_n^{w_n}$.) A COT protocol for language L allows sender S to encrypt message m "under" statement x so that receiver R gets m only if R holds a witness for membership of x in L, while S learns nothing. A *private* COT for L hides not only message m but also statement x from any R that does not know a witness for x in L.

1 Introduction

Authentication Privacy and Mutual Authentication. It seems evident that if party A authenticates itself to some verifier then A must necessarily reveal some information about itself in the process. At the minimum, an authentication protocol seemingly needs to reveal that A is credentialed by a given Certification Authority (CA), because the goal of (policy-based) uni-directional authentication is to let any verifier learn whether A holds valid credentials from a given CA. However, in the case of a *mutual* authentication, where A authenticates itself to B as certified by a CA of B's choice but it cares to do so only if B is itself appropriately certified by the CA of A's choice, we can ask whether we can protect each party's privacy fully, including its affiliation with particular CA, against any entity which does not satisfy this party's authentication policy. In other words, we ask for a protocol which mimics the following ideal private mutual authentication functionality, or an *(unlinkable) Secret Handshake* (SH): A and B input their certificates and authentication policies, and the functionality returns 1 if A's certificate matches B's policy and B's certificate matches

A's policy, and 0 otherwise. To be practical, SH scheme should match the performance of a standard non-private PKI system (or a Group Signature Scheme): The certificate of each user should be short and re-usable, the CA's public key should be short, authentication protocol should take $O(1)$ rounds and public-key operations, revocation information should be at most linear in the number of revoked players, and the scheme should support escrow, i.e. protocol participants should be efficiently traceable by a trusted party from protocol transcripts.

Applications of private mutual authentication range from peer-to-peer groups to law-enforcement agencies who might be concerned with their privacy in the sense of not wanting to publicly advertise the fact of their membership in a given group. They might want to do this due to privacy concerns, e.g. in the case of parties or clubs, to business concerns, e.g. a company who wants its employees or trading partners to be unrecognizable by competition, or due to security concerns, e.g. in the case of members of some law-enforcement agency whose safety is enhanced if their membership in the agency is not advertised. Secret Handshakes allow members in any such group to constrain the dissemination of the fact of their group membership only to other group members. Any group member can still identify other members by engaging them in an authentication protocol, but using privacy escrow and revocation mechanisms a group can revoke any member who poses a privacy risk to others.

Related Work. There exist efficient *linkable* Secret Handshakes which hide the participants' policy and source of certificates [BDS$^+$03, CJT04, JKT07, JL08], but they publicly reveal unique tokens assigned to each certificate, thus making protocol instances executed by the same party linkable. SH's can also be thought of as a bidirectional counterpart to private *uni-directional* authentication, i.e. identity-escrow [KP97], group signatures [CvH91], or unlinkable credentials [CL01], but uni-directional authentication unconditionally reveals prover's affiliation to any verifier. An SH scheme without key escrow would be implied by *key-private* broadcast encryption, whose ciphertext cannot be linked to the broadcast encryption key. The two parties could then privately establish an authenticated key by encrypting nonces under broadcast encryption keys associated with their CA's. However, the ciphertexts of existing broadcast encryption schemes, e.g. [NNL02, BGW05], can be easily linked to the revocation lists corresponding to their encryption keys. The key-private broadcast encryption of [BBW06] has limited applicability because its ciphertext size is linear in group size, while the key-private broadcast encryption of [JL07] is stateful, and the corresponding SH scheme works only if group members have roughly synchronized certificate revocation lists. (Private) attribute-based encryption [GPSW06, BSW07] allows the sender to encrypt a nonce so that it can be decrypted only by a holder of a certificate for specified attributes issued by some public key, and the ciphertext can hide this attribute from anyone who does not have the corresponding certificate. However, the ciphertext in these schemes hides only the attributes and not the public key that issues the certificates.

Our Contributions. Our first contribution is the first practical private bi-directional authentication scheme a.k.a. an *(unlinkable) Secret Handshake* (SH), and the first practical private envelope scheme, a.k.a. an *Anonymous Credential* scheme (AC), i.e. an envelope scheme with the privacy properties corresponding to SH's. Namely, the receiver can recover an encrypted message if and only if its certificate matches sender's authorization policy, but the protocol hides the sender's policy from any un-authorized receiver and it hides all information about the receiver from the sender. All our schemes support certificate revocation and privacy escrow, i.e. the group manager can recover otherwise hidden identity of protocol participants from protocol transcripts. Our SH protocol has $O(1)$ communication rounds (3 in ROM) and requires about 40 exponentiations and 6 bilinear maps per player, with additional $2r$ bilinear maps if r is the size of the revocation list, and our AC scheme has twice smaller costs because it is actually just a one-sided version of our SH scheme.

Our technical contribution is an enabling tool of our SH and AC schemes, a family of efficient *Conditional Oblivious Transfer* (COT) protocols for certain cryptographically useful class of relations. A COT protocol *for relation* $\mathcal{R}$ is a protocol between a sender S and a receiver R, which allows S, running on input a statement x and a message m, to disclose m to R if and only if R holds a *witness* for x in the language associated with relation $\mathcal{R}$, i.e. a string w s.t. $(x, w) \in \mathcal{R}$. The protocol is oblivious in the sense that S does not learn anything, not even whether there *exists* w which is a valid witness for sender's statement x. We call such COT protocol *private* if it also hides S's statement x from any receiver who does not hold a valid witness for x. COT is implied by secure two-party computation [Yao86], but it was introduced as a primitive in [COR99], extended to private COT in [Cre00], and later considered e.g. in [AIR01, BK04, LL07]. All these works used slightly different terminology than ours, calling inputs x and w just bitstrings and not *statement* and *witness* as we do, and they constructed COT protocols at the cost of $O(1)$ modular exponentiations for an equality relation on bitstrings [AIR01], i.e. $x = w$, and at the $O(k)$-exponentiations cost for monotonic Boolean formulas of size k [COR99, Cre00, AIR01, LL07, BK04].

We show practical COT protocols for relations that commonly appear in various cryptographic protocols e.g. group signatures, e-cash schemes, or threshold schemes. Recall that a *representation* of a group element C in bases $\mathbf{G} = (g_1, ..., g_m)$ is a vector $\mathbf{w} = (w_1, ..., w_m)$ s.t. $C = \prod_{i=1}^{m}(g_i)^{w_i}$. We exhibit two private COT protocols, one perfectly secure for the receiver and the other perfectly secure for the sender, with the computationally protected side in both protocols secure under the DDH assumption, for any relation $\mathcal{R}_{\mathsf{REP}(\Phi)}$ of the following form: The relation $\mathcal{R}_{\mathsf{REP}(\Phi)}$, for a predicate Φ on an $n \times m$ matrix $\mathbf{w}$, consists of pairs $((\mathbf{C}, \mathbf{G}), \mathbf{w})$ s.t. $\Phi(\mathbf{w}) = 1$ and the i-th row of $\mathbf{w}$ is a representation of the i-th element in $\mathbf{C}$ in bases formed by the i-th row of $\mathbf{G}$. Both protocols we propose use a single execution of a ZKPK of values in the same matrix $\mathbf{w}$ committed using the Pedersen commitment scheme [Ped91] (or its computationally-private but perfectly-binding modification) s.t. $\Phi(\mathbf{w}) = 1$. The cost of our COT protocols is the cost of the ZKPK plus about $(4 \cdot |\mathbf{w}|)$ exponentiations for either party. Note that there exist ZKPK's for various conditions on values committed in Pedersen commitments, e.g. equality or inequality of modular sums, products, or inverses, all using $O(1)$ exponentiations. Our results transform any such ZKPK proof system into a private COT protocol for the same relation, at the cost comparable to the cost of the ZKPK. Previous COT constructions do not enable efficient COT protocols for such relations, and since many cryptographic applications rely on efficiently provable relations on committed values, practical COT protocols for such relations might enable new privacy-protecting mechanisms beyond our SH and AC schemes.

Note that private COT forms an encryption counterpart to a zero-knowledge proof of knowledge: The verifier can use a COT protocol to encrypt some message m "under" a statement x, and the COT protocol ensures that the prover can decrypt m only if she holds a valid witness w for x. However, private COT's can enable higher level that what is zero-knowledge proofs achieve: Consider a server who wants to grant access to some resource m to a client if and only if the client's credential cert satisfies the sender's authorization policy Pol, i.e. $\mathsf{Ver}(\mathsf{Pol}, \mathsf{cert}) = 1$. If a client proves in zero knowledge that $\mathsf{Ver}(\mathsf{Pol}, \mathsf{cert}) = 1$, this reveals the fact that the client holds a certificate which satisfies policy Pol to *any* party who engages the client in this zero-knowledge proof as a verifier. Moreover, the server who engages in a proof system as a verifier on its statement Pol, even if this proof system is zero-knowledge, might also end up revealing this statement to any party, whether or not this party holds a valid witness for this statement. In contrast, the privacy of both parties is protected if the server sends m to the client in a private COT protocol for relation Ver. Thus private COT protocols for relation Ver would enable (fully) private envelopes, and a bi-directional version of this envelope would make a private authentication scheme, and in particular this is how our AC and SH schemes are constructed.

Organization. We start with a technical roadmap in Section 2. We set notation in Section 3. We define private COT in Section 4. In Section 5 we construct a private COT for relations on discrete logarithm representations with perfect security for the receiver. (For lack of space we have to omit from these proceedings our alternative protocol with perfect security for the sender.) In Section 6 we define SH schemes and construct such scheme on the basis of a group signature by Boneh and Shacham [BS04] and a COT protocol like that of Section 5.

2 Technical Roadmap

We construct an unlinkable secret handshake using a group signature scheme and a COT protocol on an appropriate relation. One possible way of doing this could be as follows. Party A issues a group signature on a challenge message, and B sends a nonce to A via a private COT protocol, s.t. A receives it if and only if A's commitment opens to a group signature which verifies under the key specified in B's authentication policy. Then the roles of the two parties are reversed: B creates and commits to its signature, and A sends its nonce to B via the COT protocol on the same condition applied to B's commitment and A's authentication policy. The first technical challenge lies in handling revocations: It's not clear how to check whether a signature is issued by some revoked member when the signature is hidden behind the commitment, unless the signer (receiver in the COT protocol) also attaches a proof that the committed signature is *not* issued by anyone in the *receiver's* revocation list. This seems hard because the revocation list assumed by each party should also be hidden, or otherwise the affiliation of that party is immediately revealed.

To avoid the problem caused with revocation we turn to the group signature (GS) scheme with verifier-local revocation (VLR) introduced by Boneh and Shacham [BS04]. In a VLR-GS scheme the signer's certificate consists of two parts: The first is a random revocation token, unrelated to the group's public key, and the second is essentially a group manager's signature on this revocation token. In the VLR-GS scheme the signer first commits to its token using a commitment scheme which is *unlinkable* without the knowledge of the committed token, but is *traceable* given the token. (This latter property enables efficient revocation.) The group signature then consists of this committed token and a Zero-Knowledge Proof of Knowledge (ZKPK) of group manager's signature on this committed token, made non-interactive using the Fiat-Shamir heuristic.

We construct an unlinkable SH scheme using the same components of the VLR-GS scheme, but replacing the above NI-ZKPK proof with a private COT scheme for the same relation. Namely party A commits to its token, B uses the traceability procedure to check if the committed token has not been revoked, and if the check passes then B sends a nonce to A via a COT protocol s.t. A receives the nonce if and only if A has a group manager's signature on the committed token, and then the roles are reversed. The reason this yields an efficient SH construction when the VLR-GS scheme is instantiated with the scheme of [BS04] is that the relation involved in the above COT protocol belongs to the class $\mathcal{R}_{\mathsf{REP}(\Phi)}$ of relations on discrete-log representations satisfying some arithmetic constraints. In other words, the commitment to a token and the group public key can be represented as a vector $\mathbf{C}$ and a matrix $\mathbf{G}$ of group elements, while the decommitment and the group manager's signature on the committed token form a matrix of exponents $\mathbf{w}$ s.t. $\mathbf{w}$ satisfies certain set of arithmetic constraints Φ *and* each row of $\mathbf{w}$ is a discrete-log representation of a corresponding element in $\mathbf{C}$ in a vector of bases form by the same row of $\mathbf{G}$.

Technically, the security argument for the above SH scheme follows easily from the unforgeability of the group manager's signatures *if* the COT protocol guarantees extractability of a witness for the receiver's statement from a receiver which tells some information about the transferred message: In such case an adversary which breaks the security of the authentication scheme immediately implies efficient computation of a forgery of the group manager's signature, since the witness to the sender's statement must be a valid signature on an unrevoked token, and an unrevoked token is an unsigned message from adversary's point of view. A privacy argument for this SH scheme will be similarly aided if the

COT protocol also guarantees extractability of a witness from a receiver which tells any information about the sender's statement. This is why the security and privacy notion we give for a COT protocol in Section 4 requires extraction of inputs from a "successful" receiver.

Finally, we sketch our COT construction for relation $\mathcal{R}_{\mathsf{REP}(\Phi)}$. The receiver cannot just run a ZKPK of $\mathbf{w}$ s.t. the arithmetic constraint Φ is satisfied and $\mathbf{w}$ is the representation of $\mathbf{C}$ in bases $\mathbf{G}$, where $(\mathbf{C}, \mathbf{G})$ is the statement assumed by the receiver, because this would reveal this part of the receiver's inputs to any sender, and in particular it could reveal the CA who issued the receiver's certificate. Instead, the receiver can independently commit to vector $\mathbf{w}$ and perform a zero-knowledge proof of knowledge of a committed $\mathbf{w}$ which satisfies constraint Φ. This protects all information about $\mathbf{w}$ (except that $\Phi(\mathbf{w}) = 1$, but this is presumably true of any party engaging in this protocol) and it also ensures efficient extraction of some $\mathbf{w}$ s.t. $\Phi(\mathbf{w}) = 1$ from any malicious receiver. If the receiver's proof verifies then the sender follows an encryption-like procedure - somewhat reminiscent of Cramer-Shoup's projective hash [CS01] – which transfers sender's message M to the receiver but ensures that the receiver gets no information about either the sender's message M or its statement $(\mathbf{C}, \mathbf{G})$ unless the committed matrix $\mathbf{w}$ is a representation of $\mathbf{C}$ in bases $\mathbf{G}$. Looking ahead, in the COT protocol of Figure 2 this additional commitment to $\mathbf{w}$ is denoted $\mathbf{D}$, and the encryption-like procedure outputs $\mathbf{E}$, $\mathbf{F}$, $\hat{K}$ on inputs $\mathbf{C}, \mathbf{G}, \mathbf{D}$ and M.

3 Cryptographic Setting and Notation

Throughout the paper we assume that $\mathbb{G}$ is a multiplicative group of prime order q, and that g is its generator. Our security statements in section 5 assume an exact security version of the DDH assumption, i.e. we say that DDH is (t, ϵ)-hard in group $\mathbb{G}$ if any t-time algorithm $\mathcal{A}$ has at most ϵ advantage in distinguishing distributions $\{(g, g^a, g^b, g^{ab})\}_{a,b \leftarrow_R \mathbb{Z}_q}$ and $\{(g, g^a, g^b, g^c)\}_{a,b,c \leftarrow_R \mathbb{Z}_q}$. If $\mathcal{A}$ is a probabilistic algorithm then $A(x; r)$ denotes an output of A on x and random tape r. We use bold letters to denote vectors or matrices. We write $\mathbf{w} \in S^{n \times m}$ to denote a matrix $\mathbf{w}$ with dimensions $n \times m$ and elements in set S. We use $\mathbf{w}[i, j]$ to designate an element in the i-th row and j-th column of $\mathbf{w}$.

4 Definition of Private Conditional Oblivious Transfer

A COT protocol for message space $\mathcal{M}$ and relation $\mathcal{R}$ (and the language $L_{\mathcal{R}}$ implied by $\mathcal{R}$ as well as an implicit universe of "statement-looking" strings $\mathcal{U}_{\mathcal{R}} \supseteq L_{\mathcal{R}}$) consists of two probabilistic interactive algorithms S and R, which execute on S's private inputs a message M in $\mathcal{M}$ and a bitstring x, and on R's private input a bitstring w. At the end of the interaction, R outputs message M if and only if $(x, w) \in \mathcal{R}$, and S has no output. (See Figure 1.) A COT protocol must meet the following basic properties:

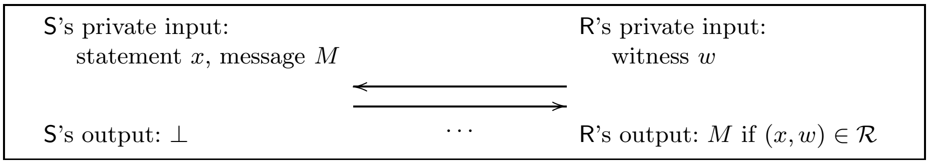


Fig. 1. Functionality of a COT scheme for relation $\mathcal{R}$ between sender S and receiver R

Definition 1 (Completeness). *A COT protocol for relation $\mathcal{R}$ and message space $\mathcal{M}$ is* complete *if for any $(x, w) \in \mathcal{R}$ and any $M \in \mathcal{M}$, at the end of the interaction between $\mathsf{S}(x, M)$ and $\mathsf{R}(w)$, R outputs M.*

Definition 2 (Security). *A COT protocol for relation $\mathcal{R}$ and message space $\mathcal{M}$ is* (t, ϵ)-secure *if for any $x \notin L_{\mathcal{R}}$, any $M_0, M_1 \in \mathcal{M}$, any t-time algorithm $\mathcal{A}$, and any auxiliary information z,*

$$\left| Pr[\mathcal{A}^{\mathsf{S}(x,M_0)}(x, M_0, M_1, z) = 1] - Pr[\mathcal{A}^{\mathsf{S}(x,M_1)}(x, M_0, M_1, z) = 1] \right| \leq \epsilon$$

where the probabilities are taken over the randomness of $\mathcal{A}$ and S.

Definition 3 (Receiver Privacy). *A COT protocol for relation $\mathcal{R}$ is (t, ϵ)-*receiver private *if for any t-time algorithm $\mathcal{A}$, any w_0, w_1, and any auxiliary information z,*

$$\left| Pr[\mathcal{A}^{\mathsf{R}(w_0)}(w_0, w_1, z) = 1] - Pr[\mathcal{A}^{\mathsf{R}(w_1)}(w_0, w_1, z) = 1] \right| \leq \epsilon$$

where the probabilities are taken over the randomness of $\mathcal{A}$ and R.

However, the above security property has several limitations. First, it allows the protocol to reveal sender's message to any receiver if the sender's statement x *is* in the language. A more useful notion would require that the message is revealed only to the receiver who holds a valid witness for x. This requirement can be captured via extractability, i.e. if the receiver distinguishes the execution of $S(x, M_0)$ and $S(x, M_1)$ then a witness w for x can be efficiently extracted from this receiver. Moreover, a *private* COT protocol should protect sender's statement in a similar way, i.e. if the receiver distinguishes the execution of $S(x_0, M)$ and $S(x_1, M)$ then a witness for *either* x_0 *or* x_1 can be extracted from this receiver. We capture both of these properties in a notion of *strong security and sender privacy* defined below. Technically, we define this notion in terms of distinguishing between a "real" sender $S(x, M)$ and a "simulated" sender $S(x', M')$ which runs on any statement x' and a *random* message M': An adversary can distinguish between these two only if a witness for x in $L_{\mathcal{R}}$ can be extracted from this adversary. Note that this notion implies the intuitive security and sender privacy properties discussed above. Moreover, this notion is convenient for arguing security of applications of a private COT because it implies that if an adversary distinguishes real and simulated protocols then the reduction can extract a witness for the real sender's statement.

Definition 4 (Strong Security and Sender Privacy). *A COT protocol for relation $\mathcal{R}$, statement universe $\mathcal{U}$, and message space $\mathcal{M}$ is* strongly secure and sender private with soundness error δ *if there exists an efficient extractor algorithm* Ext *and a polynomial $p(\cdot)$ s.t. for any $x, x' \in \mathcal{U}$, any $M \in \mathcal{M}$, any efficient probabilistic algorithm $\mathcal{A}$, any auxiliary information z (w.l.o.g. z contains x, x', M), and any randomness vector r, if*

$$\epsilon_{\mathcal{A},z,r} \triangleq \left| Pr_{\{\$_\mathsf{S}\}}[\mathcal{A}^{\mathsf{S}(x,M)}(z;r) = 1] - Pr_{\{\$_\mathsf{S}, M' \leftarrow_R \mathcal{M}\}}[\mathcal{A}^{\mathsf{S}(x',M')}(z;r) = 1] \right| > \delta$$

then

$$Pr_{\{\$_\mathsf{Ext}\}}\left[(x,w) \in \mathcal{R} |\ w \leftarrow \mathsf{Ext}^{\mathcal{A}(z;r)}\right] \geq p(\epsilon_{\mathcal{A},z,r} - \delta)$$

where $\$_\mathsf{S}$ and $\$_\mathsf{Ext}$ are the randomness of S *and* Ext *respectively.*
In concrete security terms, we call a COT protocol $(t, t_{ext}, q_{ext}, d, e)$-strongly secure and sender private with soundness error δ *if the above requirement is satisfied for any t-time adversary $\mathcal{A}$, for polynomial $p(\epsilon) = d\epsilon^e$, and for algorithm* Ext *running in time t_{ext} and making at most q_{ext} calls to $\mathcal{A}$.*

5 Private COT Protocol for Relations on Representations

We give two constructions of a private COT protocol for any relation on so-called representations of group elements. Our first construction relies on a witness-indistinguishable proof of knowledge (WIPoK) for the same relation on values committed using Pedersen commitment scheme [Ped91]. The second construction needs a Strong WIPoK for the same relation on values committed in the following simple perfectly binding but computationally hiding commitment scheme: $Com_{g,h,y}(m) = (g^r, y^r h^m)$. Security and sender privacy of the first COT protocol construction relies on the DDH assumption and the strong soundness of the WIPoK proof system, while receiver privacy relies on witness-indistinguishability of the WIPoK. For the second construction, security and sender privacy relies on strong soundness of the SWIPoK, while receiver privacy relies on DDH assumption and strong witness-indistinguishability of the SWIPoK. We show the first construction below, while for lack of space we relegate our second construction to the full version of this paper.

Let $\mathbb{G}$ be a multiplicative group of prime order q. A *representation* of group element $C \in \mathbb{G}$ in bases $(g_1, ..., g_n) \in \mathbb{G}^n$ is any vector $(w_1, ..., w_n) \in (\mathbb{Z}_q)^n$ s.t. $C = \prod_{i=1}^n (g_i)^{w_i}$. Let Φ be a relation on sets of $n \times m$ elements in $\mathbb{Z}_q$, i.e. $\Phi : (\mathbb{Z}_q)^{n \times m} \to \{0,1\}$. Assume that Φ is satisfiable, i.e. there exists $\mathbf{w}$ s.t. $\Phi(\mathbf{w}) = 1$. Moreover, assume that there exists an efficient procedure to find (any) $\mathbf{w}$, s.t. $\Phi(\mathbf{w}) = 1$.

We define a language $\mathsf{REP}(\Phi)$ as a set of pairs $(\mathbf{G}, \mathbf{C})$,

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}[1,1],\ \mathbf{G}[1,2],\ \ldots,\ \mathbf{G}[1,m] \\ \mathbf{G}[2,1],\ \mathbf{G}[2,2],\ \ldots,\ \mathbf{G}[2,m] \\ \ldots \\ \mathbf{G}[n,1],\ \mathbf{G}[n,2],\ \ldots,\ \mathbf{G}[n,m] \end{pmatrix} \in \mathbb{G}^{n \times m} \quad , \quad \mathbf{C} = \begin{pmatrix} \mathbf{C}[1] \\ \mathbf{C}[2] \\ \ldots \\ \mathbf{C}[n] \end{pmatrix} \in \mathbb{G}^n$$

s.t. $\exists\ \mathbf{w} \in (\mathbb{Z}_q)^{n\times m}$ s.t. $\Phi(\mathbf{w}) = 1$ and $\mathbf{C}[i] = \prod_{j=1}^{m}(\mathbf{G}[i,j])^{\mathbf{w}[i,j]}$ for all $i \in [1..n]$.

$\mathcal{R}_{\mathsf{REP}(\Phi)}$ is a relation corresponding to this language, i.e. set of pairs $((\mathbf{G},\mathbf{C}),\ \mathbf{w})$ which satisfy the above conditions, and $\mathcal{U}_{\mathsf{REP}(\Phi)}$ includes all $(\mathbf{G},\mathbf{C}) \in \mathbb{G}^{n\times m}\times\mathbb{G}^n$.

Construction of Private COT for Relation $\mathcal{R}_{\mathsf{REP}(\Phi)}$. We construct a private COT protocol for $\mathcal{R}_{\mathsf{REP}(\Phi)}$ given a witness-indistinguish- able proof of knowledge (WIPoK) for the following language:

$$\mathsf{PedREP}_{g,h}(\Phi) \triangleq \left\{ \begin{array}{l} \mathbf{D} \in \mathbb{G}^{n\times m} \text{ s.t. } \exists \mathbf{w},\mathbf{r} \in (\mathbb{Z}_q)^{n\times m} \text{ s.t. } \Phi(\mathbf{w}) = 1 \\ \qquad\qquad \text{and } \forall_{(i,j)\in[1..n]\times[1..m]}\ \mathbf{D}[i,j] = g^{\mathbf{w}[i,j]}\cdot h^{\mathbf{r}[i,j]} \end{array} \right\}$$

Note that PedREP is a trivial language, i.e. every $\mathbf{D} \in \mathbb{G}^{n\times m}$ is in PedREP. However, we require a (non-trivial) proof of knowledge of $(\mathbf{w},\mathbf{r})$, given $\mathbf{D}$, s.t. $(\mathbf{D},(\mathbf{w},\mathbf{r})) \in \mathcal{R}_{\mathsf{PedREP}}$ where $\mathcal{R}_{\mathsf{PedREP}}$ is a relation corresponding to this language, i.e. set of pairs $(\mathbf{D},(\mathbf{w},\ \mathbf{r}))$ which satisfy above conditions.

Practical ZKPK (and WIPoK) proofs exist for languages $\mathsf{PedREP}_{g,h}(\Phi)$ for many useful constraints Φ. First, note that there exist efficient HVZKPK proof systems with special HVZK and properties for many constraints involving values committed in Pedersen commitment, e.g. linear equations, i.e. $\phi(\mathbf{w}) = 1$ if $a_1\mathbf{w}[i_1,j_1] + a_2\mathbf{w}[i_2,j_2] = a_3\mathbf{w}[i_3,j_3]$ for some $i_1,i_2,i_3 \in [1..n]$ and $j_1,j_2,j_3 \in [1..m]$ and constants a_1,a_2,a_3, or quadratic equations, i.e. $\mathbf{w}[i_1,j_1] = \mathbf{w}[i_2,j_2]\cdot\mathbf{w}[i_3,j_3]$ (see e.g. [CM99]), a "less than" relation [Bou00], i.e. $\phi(\mathbf{w}) = 1$ iff $\mathbf{w}[i_1,j_1] \leq \mathbf{w}[i_2,j_2]$, or an inequality relation, i.e. $\phi(\mathbf{w}) = 1$ iff $\mathbf{w}[i_1,j_1] \neq \mathbf{w}[i_2,j_2]$. Secondly, by results of[CDS94], such HVZKPK's can be "compiled" into an efficient HVZKPK's for any constraint Φ formed by conjunctions and disjunctions of such constraints. Finally, all such HVZKPK proof systems can be compiled, with negligible overhead, into ZKPK proof systems, non-interactive in ROM model (using Fiat-Shamir heuristic), 3-round in CRS model [Dam00], or 5-round in the standard model [MP03].

The protocol proceeds given group $\mathbb{G}$ with generator g, on sender's private inputs an instance $(\mathbf{G},\mathbf{C}) \in \mathbb{G}^{n\times m}\times\mathbb{G}^n$ and a message $M \in \mathbb{G}$, and on receiver's private input $\mathbf{w}$. First the sender S sends to the receiver R a random h in $\mathbb{G}\setminus\{1\}$. R aborts if $h = 1$. If $\Phi(\mathbf{w}) \neq 1$, then R picks $\mathbf{w}'$, s.t. $\Phi(\mathbf{w}') = 1$, and sets $\mathbf{w} \leftarrow \mathbf{w}'$. Then R sends to S Pedersen commitments to all $\mathbf{w}[i,j]$'s in $\mathbf{w}$: picks $\mathbf{r} \leftarrow_R \mathbb{G}^{n\times m}$, creates $\mathbf{D}$, s.t. $\mathbf{D}[i,j] = g^{\mathbf{w}[i,j]}h^{\mathbf{r}[i,j]}$, and proves using the WIPoK proof system for $\mathsf{PedREP}_{g,h}(\Phi)$ that the committed values $\mathbf{w}$ satisfy the Φ relation, i.e. that $(\mathbf{D},(\mathbf{w},\mathbf{r})) \in \mathcal{R}_{\mathsf{PedREP}}$. If R passes the proof, S uses the instance $(\mathbf{G},\mathbf{C})$ and commitments $\mathbf{D}$ to encrypt M as follows: S picks random s_i's in $\mathbb{Z}_q$ for every $i \in [1..n]$ and random $t_{i,j}$'s in $\mathbb{Z}_q$ for every $(i,j) \in [1..n]\times[1..m]$, and sends to R the sets $\mathbf{E},\mathbf{F}$ of ciphertexts $\mathbf{E}[i,j]$ and $\mathbf{F}[i,j]$,

$$\forall_{(i,j)\in[1..n]\times[1..m]} \quad \mathbf{E}[i,j] = (\mathbf{G}[i,j])^{s_i}g^{t_{i,j}} \text{ and } \mathbf{F}[i,j] = h^{t_{i,j}}$$

together with value $\hat{K} = \prod_{i=1}^{n} K_i \cdot M$, where

$$\forall_{i\in[1..n]} \quad K_i = (\mathbf{C}[i])^{s_i}\cdot\prod_{j=1}^{m}(\mathbf{D}[i,j])^{t_{i,j}}$$

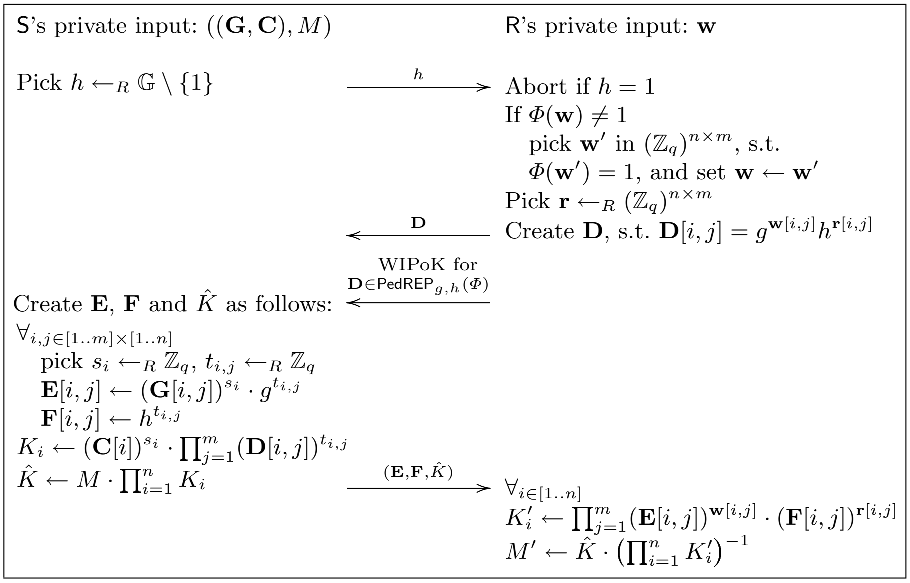

S's private input: $((\mathbf{G},\mathbf{C}),M)$
R's private input: $\mathbf{w}$
Pick $h \leftarrow_R \mathbb{G} \setminus \{1\}$
$\xrightarrow{h}$
Abort if $h = 1$
If $\Phi(\mathbf{w}) \neq 1$
pick $\mathbf{w}'$ in $(\mathbb{Z}_q)^{n\times m}$, s.t.
$\Phi(\mathbf{w}') = 1$, and set $\mathbf{w} \leftarrow \mathbf{w}'$
Pick $\mathbf{r} \leftarrow_R (\mathbb{Z}_q)^{n\times m}$
$\xleftarrow{\mathbf{D}}$
Create $\mathbf{D}$, s.t. $\mathbf{D}[i,j] = g^{\mathbf{w}[i,j]}h^{\mathbf{r}[i,j]}$
WIPoK for
$\xleftarrow{\mathbf{D}\in\mathsf{PedREP}_{g,h}(\Phi)}$
Create $\mathbf{E}$, $\mathbf{F}$ and $\hat{K}$ as follows:
$\forall_{i,j\in[1..m]\times[1..n]}$
pick $s_i \leftarrow_R \mathbb{Z}_q$, $t_{i,j} \leftarrow_R \mathbb{Z}_q$
$\mathbf{E}[i,j] \leftarrow (\mathbf{G}[i,j])^{s_i} \cdot g^{t_{i,j}}$
$\mathbf{F}[i,j] \leftarrow h^{t_{i,j}}$
$K_i \leftarrow (\mathbf{C}[i])^{s_i} \cdot \prod_{j=1}^{m}(\mathbf{D}[i,j])^{t_{i,j}}$
$\hat{K} \leftarrow M \cdot \prod_{i=1}^{n} K_i$
$\xrightarrow{(\mathbf{E},\mathbf{F},\hat{K})}$
$\forall_{i\in[1..n]}$
$K_i' \leftarrow \prod_{j=1}^{m}(\mathbf{E}[i,j])^{\mathbf{w}[i,j]} \cdot (\mathbf{F}[i,j])^{\mathbf{r}[i,j]}$
$M' \leftarrow \hat{K} \cdot \left(\prod_{i=1}^{n} K_i'\right)^{-1}$

Fig. 2. COT Protocol for Relation $\mathcal{R}_{\mathsf{REP}(\Phi)}$ (and Message Space $\mathbb{G}$)

Finally R decrypts $(\mathbf{E},\mathbf{F},\hat{K})$ as $M' \leftarrow \hat{K} \cdot K_1' \cdot ... \cdot K_n'$, where

$$\forall_{i\in[1..n]} \quad K_i' = \prod_{j=1}^{m}(\mathbf{E}[i,j])^{\mathbf{w}[i,j]} \cdot (\mathbf{F}[i,j])^{\mathbf{r}[i,j]}$$

Theorem 1. *If DDH problem is (t_{ddh},ϵ_{ddh})-hard, and the proof system in the construction is (t_{ext},q_{ext},d,e)-strongly-sound with soundness error δ, then the above construction of COT protocol for $\mathcal{R}_{\mathsf{REP}(\Phi)}$ is $(t_{adv},t_{ext},q_{ext},d',e',)$-strongly-secure-and-sender-private, with soundness error δ', where $t_{adv} = t_{ddh} - O(nm)t_{exp}$, $d' = \frac{1}{2^{e+1}}d$, $e' = e+1$, $\delta' = 2\delta + 4nm\epsilon$, and t_{exp} is the time for one exponentiation operation.*

Proof sketch: First, by splitting lemma, if adversary $\mathcal{A}$ has $\epsilon_{\mathcal{A}}$ advantage in distinguishing the real game from the random game, i.e., between $\mathsf{S}(x,M)$ and $\mathsf{S}(x',M')$ for random M' in $\mathbb{G}$, then for $\epsilon_{\mathcal{A}}/2$ portion of the h values sent by S in the first round, $\mathcal{A}$ has $\epsilon_{\mathcal{A}}/2$ advantage in distinguishing the two games, where in both games h is fixed to this chosen value. Suppose h sent by S is from this portion. By strong soundness of the proof system, a pair $(\mathbf{w},\mathbf{r})$ can be extracted from $\mathcal{A}$, s.t. $\mathbf{D}[i,j] = g^{\mathbf{w}[i,j]} \cdot h^{\mathbf{r}[i,j]}$. So what remains to argue is that the extracted $\mathbf{w}$ is the witness for the real sender $\mathsf{S}(x,M)$'s statement $x = (\mathbf{G},\mathbf{C})$. Note that (1) for each (i,j) pair, $(\mathbf{F}[i,j],\mathbf{E}[i,j])$ is an ElGamal encryption of element $(\mathbf{G}[i,j])^{s_i}$ under "key" h. Therefore by DDH assumption $(\mathbf{E},\mathbf{F})$ is indistinguishable from $U_{\mathbb{G}^{n\times m}\times\mathbb{G}^{n\times m}}$, where U_G denotes uniform distribution over group G. Hence $(\mathbf{E}',\mathbf{F}',\hat{K}')$ sent by $\mathsf{S}(x',M')$ in the random game is indistinguishable from $U_{\mathbb{G}^{n\times m}\times\mathbb{G}^{n\times m}\times\mathbb{G}}$ because M' is random in $\mathbb{G}$;

and (2) $K_i = \mathbf{C}[i]^{s_i} \cdot \prod_{j=1}^{m}(\mathbf{D}[i,j])^{t_{i,j}}$ computed by $\mathsf{S}((\mathbf{G},\mathbf{C}),M)$ is indeed $\prod_{j=1}^{m}((\mathbf{E}[i,j])^{\mathbf{w}[i,j]} \cdot (\mathbf{F}[i,j])^{\mathbf{r}[i,j]}) \cdot (\mathbf{C}[i] \cdot (\prod_{j=1}^{n}(\mathbf{G}[i,j])^{-\mathbf{w}[i,j]}))^{s_i}$. If $\mathbf{w}$ is *not* the witness for $(\mathbf{G},\mathbf{C})$, then there exists at least one i, s.t. $\mathbf{C}[i] \neq \prod_{j=1}^{n}(\mathbf{G}[i,j])^{\mathbf{w}[i,j]}$. Then because s_i is random in $\mathbb{Z}_q$, K_i is random in $\mathbb{G}$ and so is $\hat{K}$. Hence $(\mathbf{E},\mathbf{F},\hat{K})$ sent by the real sender $\mathsf{S}((\mathbf{G},\mathbf{C}),M)$ is also indistinguishable from $U_{\mathbb{G}^{n\times m}\times\mathbb{G}^{n\times m}\times\mathbb{G}}$. Therefore, the only way $\mathcal{A}$ can tell a difference between the real and random games is either by breaking the DDH assumption or by feeding the "correct" witness for the real sender $\mathsf{S}(x,M)$'s statement and then the extractor can extract it with large enough probability. □

The proof of the following theorem is simple, so we omit it for lack of space:

Theorem 2. *If the proof system for* PedREP *is (t,ϵ)-witness-indistinguishable, then the constructed COT protocol is (t,ϵ)-receiver-private.*

6 Construction of Unlinkable Secret Handshake Scheme

We construct an unlinkable SH scheme from so-called "Verifier-Local Revocable" Group Signatures (VLR GS), introduced and realized under Strong Diffie-Hellman and Decisional Linear assumptions by Boneh and Shacham [BS04]. Below we define unlinkable secret handshakes, specify the properties of a VLR-GS scheme that are useful to us, and show a construction of an SH scheme using such VLR-GS scheme and private COT protocol.

6.1 (Unlinkable) Secret Handshakes: Definition

An (Unlinkable) Secret Handshake Scheme (SH) is an authenticated key exchange protocol which operates in an environment with many groups, each managed by some group manager GM, and N users $P_1, ..., P_N$, each of which can be a member of several groups. Each GM plays a role of the Certificate Authority for its group, issuing certificates to any user it wants to admit to its group, and publishing revocation tokens for any user it wants to revoke from it. An SH scheme consists of three algorithms Setup, KGen, and Trace, and an interactive procedure Handshake, s.t.

- Setup on security parameter κ outputs parameters par (and key space $\mathcal{K}$).
- KGen, executed by a group manager GM on input par, outputs a group public key gpk and a vector of user keys $\mathbf{usk} = (\mathbf{usk}[1], \mathbf{usk}[2], \ldots, \mathbf{usk}[N])$ and revocation tokens $\mathbf{urt} = (\mathbf{urt}[1], \mathbf{urt}[2], \ldots, \mathbf{urt}[N])$. For notational simplicity we assume that user P_i is given key $\mathbf{usk}[i]$ for every group it belongs to.
- Handshake is an interactive protocol between two users, where each P_i runs on its private inputs $(\mathbf{usk}[i], \mathsf{gpk})$, and outputs a pair (k, tr) where $k \in \mathcal{K}$ is a key material to be used for subsequent secure communication with the protocol counterparty and tr is an escrow of that counterparty's identity.
- Trace, on inputs $(\mathsf{tr}, \mathbf{urt}[i])$ outputs 1 if tr is linked to $\mathbf{usk}[i]$, 0 otherwise.

Remark on Trace usage: Algorithm Trace has two uses: First, it can be used by the group manager to de-escrow the identity of a player involved in any protocol instance. Second, the intended usage of the above $\mathsf{Handshake}$ protocol, in which player P_i always outputs some (k, tr) pair, is to be followed by a verification that tr does not open to any revocation tokens included in the revocation list. If it does, P_i throws away the created key k.

Remark on privacy of revoked users: The revocation tokens are kept secret by the group manager, and published only to revoke a given player from a group. Therefore all past transcripts of a given user can be linked to this user, via the Trace algorithm, once the user is revoked. Such privacy limitation is a feature the verifier-local revocation group signatures [BS04]. A stronger privacy model, where past transcripts of revoked users remain private, can be supported by group signature schemes using accumulators, e.g. [CL01, BBS04]. It is an open question whether similar privacy can be efficiently achieved by an unlinkable SH scheme. (The major difficulty stems from the fact that two communicating players might assume different revocation epochs, and hence run the SH protocol on incompatible accumulators.)

Properties of SH Scheme. An SH scheme must meet the following properties:

Completeness: For every par output by Setup and every $(\mathsf{gpk}, \mathbf{usk}, \mathbf{urt})$ output by KGen on par, if any two players P_i and P_j honestly execute the $\mathsf{Handshake}$ protocol with inputs $(\mathbf{usk}[i], \mathsf{gpk})$ and $(\mathbf{usk}[j], \mathsf{gpk})$ respectively, then their respective outputs (k_i, tr_i) and (k_j, tr_j) satisfy $k_i = k_j$.[1] (It will follow from the security definition below that also $\mathsf{Trace}(\mathsf{tr}_j, \mathbf{urt}[i]) = 1$ and $\mathsf{Trace}(\mathsf{tr}_i, \mathbf{urt}[j]) = 1$.)

Security (Traceability): Security of an SH scheme is similar to traceability in a group signature scheme. Namely, it requires that if some player successfully authenticates itself to some player P_i then P_i's transcript of this protocol can be linked to that player's identity. Formally, security of an SH scheme is defined via the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}^{\mathsf{sec}}$, on input any par output by Setup:

– **Init**. The challenger $\mathcal{CH}^{\mathsf{sec}}$, on input par and a bit b, runs $\mathsf{KGen}(\mathsf{par})$, which defines $(\mathsf{gpk}, \mathbf{usk}, \mathbf{urt})$, sends $(\mathsf{gpk}, \mathbf{urt})$ to $\mathcal{A}$, and sets $\mathsf{Cor} \leftarrow \emptyset$.
– **Queries**. $\mathcal{A}$ can make the following queries, where each query is serviced by the challenger *sequentially*, which disallows man in the middle attacks:
  - **Handshake**(i). $\mathcal{CH}^{\mathsf{sec}}$ on this query performs the $\mathsf{Handshake}$ protocol on inputs $(\mathbf{usk}[i], \mathsf{gpk})$, interacting with $\mathcal{A}$.
  - **Corruption**(i). $\mathcal{CH}^{\mathsf{sec}}$ sends $\mathbf{usk}[i]$ to $\mathcal{A}$ and adds i to Cor.

[1] For notational simplicity, we present the completeness definition in the "symmetric" setting where two players authenticate each other if they are in the same group. However, our constructions generalize to the "asymmetric" setting, i.e. if $\mathbf{usk}[i]$ is issued under gpk_j and $\mathbf{usk}[j]$ is issued under gpk_i, then $k_i = k_j$.

  - **Challenge**(i). (Allowed only once.) $\mathcal{CH}^{\mathsf{sec}}$ acts as in the **Handshake**(i) query, but denotes its outputs as (k, tr). If $\mathsf{Trace}(\mathsf{tr}, \mathbf{urt}[j]) = 1$ for any $j \in \mathsf{Cor}$, the game stops. Else, $\mathcal{CH}^{\mathsf{sec}}$ assigns $k_0 \leftarrow k$, picks $k_1 \leftarrow_R \mathcal{K}$, and gives k_b to $\mathcal{A}$.

– **Guess**. $\mathcal{A}$ outputs b' as its guess of b.

Let $p_b = \Pr[\mathcal{A}^{\mathcal{CH}^{\mathsf{sec}}(b,\mathsf{par})}(\mathsf{par}) = 1]$, where the probability goes over the randomness of both $\mathcal{A}$ and $\mathcal{CH}^{\mathsf{sec}}$, and let $\mathsf{Adv\text{-}Sec}(\mathcal{A}, \mathsf{par}) = |p_1 - p_0|$. We say that an SH scheme is (t, q_{sh}, N, ϵ)*-secure* on parameters par if in a universe with N users, for any t-time adversary $\mathcal{A}$ making at most q_{sh} **Handshake** queries, $\mathsf{Adv\text{-}Sec}(\mathcal{A}, \mathsf{par}) \leq \epsilon$.

Upgrade to Authenticated Key Agreement: If the $\mathsf{Handshake}$ protocol meets the above notion then it should be straightforward to convert it to an Authenticated Key Exchange (AKE) protocol secure against man-in-the-middle attacks. However, since modeling of AKE protocols requires introduction of an extended formalism, e.g. [BCK01], such compilation is out of scope of this paper.

Privacy: The privacy property covers both the anonymity property of group signatures, i.e. that no one except the group manager can detect if two instances of the SH protocol are executed by the same user, together with the affiliation-hiding property of secret handshake protocols (e.g. [BDS$^+$03, CJT04]), i.e. that no one can detect which group a given player belongs to except of non-revoked members of the same group. Formally, we define privacy via the following game between an adversary $\mathcal{A}$ and the challenger $\mathcal{CH}^{\mathsf{pri}}$, on input any parameters par output by Setup:

– **Init**. The challenger $\mathcal{CH}^{\mathsf{pri}}$, on input par and a bit b, runs $\mathsf{KGen}(\mathsf{par})$ for every group G, with outputs denoted $(\mathsf{gpk}_G, \mathbf{usk}_G, \mathbf{urt}_G)$, gives gpk_G for all groups G to $\mathcal{A}$, and sets $\mathsf{Cor} \leftarrow \emptyset$ and $\mathsf{Chosen} \leftarrow \emptyset$.

– **Queries**. $\mathcal{A}$ can make the following types of queries. As in the security game, the challenger services each query sequentially:
  - **Handshake**(i, G). $\mathcal{CH}^{\mathsf{pri}}$ runs $\mathsf{Handshake}$ on $(\mathbf{usk}_G[i], \mathsf{gpk}_G)$, interacting with $\mathcal{A}$.
  - **Corruption**(i). If $i \notin \mathsf{Chosen}$ then $\mathcal{A}$ gets $\mathbf{usk}_G[i], \mathbf{urt}_G[i]$ for every G. Let $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{i\}$.
  - **Challenge**(i_0, G_0, i_1, G_1). (Allowed only once.) Set $\mathsf{Chosen} \leftarrow \{i_0, i_1\}$. If $\mathsf{Chosen} \cap \mathsf{Cor} \neq \emptyset$ then the game stops. Otherwise $\mathcal{CH}^{\mathsf{pri}}$ runs $\mathsf{Handshake}$ on input $(\mathbf{usk}_{G_b}[i_b], \mathsf{gpk}_{G_b})$, interacting with $\mathcal{A}$.

– **Guess**. $\mathcal{A}$ outputs b' as its guess of b.

Let $p_b = \Pr[\mathcal{A}^{\mathcal{CH}^{\mathsf{pri}}(b,\mathsf{par})}(\mathsf{par}) = 1]$, where the probability goes over the randomness of both $\mathcal{A}$ and $\mathcal{CH}^{\mathsf{pri}}$, and let $\mathsf{Adv\text{-}Pri}(\mathcal{A}, \mathsf{par}) = |p_1 - p_0|$. We say that a SH scheme is (t, q_{sh}, N, ϵ)*-private* on parameters par if in a universe with N users, for any t-time adversary $\mathcal{A}$ making at most q_{sh} **Handshake** queries, $\mathsf{Adv\text{-}Sec}(\mathcal{A}, \mathsf{par}) \leq \epsilon$.

6.2 Verifier-Local Revocable Group Signature (VLR-GS)

A VLR-GS scheme consists of the following algorithms: A setup procedure $\mathsf{Setup}_{\mathsf{GS}}$ which creates public parameters par, an unforgeable *certificate scheme* $\Pi_{cert} = (\mathsf{KeyGen}, \mathsf{Cert}_{\mathsf{par}}, \mathsf{Ver}_{\mathsf{par}})$, a non-interactive zero-knowledge proof for relation $\mathcal{R}_{\mathsf{AUTH}}$ which we define below, and two additional procedures $\mathsf{Com}_{\mathsf{par}}$ and $\mathsf{TraceCom}_{\mathsf{par}}$. The functionality of the certificate scheme Π_{cert} is that if $(\mathsf{sk}, \mathsf{pk})$ is an output by $\mathsf{KeyGen}(\mathsf{par})$ then each run of $\mathsf{Cert}_{\mathsf{par}}(\mathsf{sk})$ generates a new token/secret pair $(\mathsf{tk}, \mathsf{scr})$ s.t. $\mathsf{Ver}_{\mathsf{par}}(\mathsf{pk}, \mathsf{tk}, \mathsf{scr}) = 1$. To enable an efficient VLR-GS scheme, three conditions must be met:

I. The outputs of Com must be *traceable* in the sense that (1) $\mathsf{TraceCom}_{\mathsf{par}}(C, \mathsf{tk}) = \mathsf{TraceCom}_{\mathsf{par}}(C, \mathsf{tk}') = 1$ implies $\mathsf{tk} = \mathsf{tk}'$, and (2) $\mathsf{TraceCom}_{\mathsf{par}}\ (C, \mathsf{tk}) = 1$ if and only if $\exists\, r$ s.t. $C = \mathsf{Com}_{\mathsf{par}}(\mathsf{tk}; r)$, and that .

II. There must exist an efficient non-interactive ZKPK proof system for language

$$\mathsf{AUTH}(\mathsf{par}) = \left\{ \begin{array}{ll} (C, \mathsf{pk}) & \text{s.t. } \exists\,(\mathsf{tk}, \mathsf{scr}, r) \text{ s.t. } C = \mathsf{Com}_{\mathsf{par}}(\mathsf{tk}; r) \\ & \text{and } \mathsf{Ver}_{\mathsf{par}}(\mathsf{pk}, \mathsf{tk}, \mathsf{scr}) = \mathsf{accept} \end{array} \right\}$$

III. The certificate scheme Π_{cert} must be existentially unforgeable:

Definition 5. *We say that the certificate scheme* Π_{cert} *is* $(t, \hat{q}, \epsilon)$-unforgeable *on parameters* par *if for any t-time adversary* $\mathcal{A}$*, the probability of the following event is at most* ϵ*: First* $(\mathsf{sk}, \mathsf{pk})$ *is generated by* $\mathsf{KeyGen}(\mathsf{par})$*, then* $\mathsf{Cert}_{\mathsf{par}}(\mathsf{sk})$ *is executed* $\hat{q}$ *times to generate* $\hat{q}$ *token/secret pairs* $(\mathsf{tk}_i, \mathsf{scr}_i)$*, and then* $\mathcal{A}$ *on input* par*,* pk*, and* $\{\mathsf{tk}_i, \mathsf{scr}_i\}_{i=1,..,\hat{q}}$*, outputs* $(\mathsf{tk}^*, \mathsf{scr}^*)$ *s.t.* $\mathsf{Ver}_{\mathsf{par}}(\mathsf{pk}, \mathsf{tk}^*, \mathsf{scr}^*) = 1$ *and* $\mathsf{tk}^* \neq \mathsf{tk}_i$ *for all i. The probability in this experiment runs over the randomness of* $\mathcal{A}$ *and procedures* KeyGen *and* Cert*.*

Under the above conditions a VLR-GS scheme works as follows. The group public key is pk output by KeyGen, and each group member's signature key is $(\mathsf{tk}, \mathsf{scr})$ output by $\mathsf{Cert}_{\mathsf{par}}$ on the corresponding sk. A signature under group key pk consists of $C = \mathsf{Com}_{\mathsf{par}}(\mathsf{tk})$ and a non-interactive ZKPK for $(C, \mathsf{pk}) \in \mathsf{AUTH}(\mathsf{par})$. Any user can be revoked by the group manager adding the token part tk in his/her key to the CRL. A verifier then checks if $\mathsf{TraceCom}_{\mathsf{par}}(C, \mathsf{tk}) = 1$ for each tk in the CRL. However, to enable our SH construction a VLR-GS scheme must meet two more properties:

IV. Token tk in pair $(\mathsf{tk}, \mathsf{scr})$ output by $\mathsf{Cert}_{\mathsf{par}}(\mathsf{sk})$ must be uniformly distributed in some set $\mathbb{U}_t$ defined by par and independent of key sk.

V. Values C output by $\mathsf{Com}_{\mathsf{par}}(\mathsf{tk})$ hide the tk value, not in the sense of semantic security, because knowledge of tk enables linking C to tk via $\mathsf{TraceCom}$, but in the following sense:

Definition 6. *We say that the algorithm* Com *is* (t, q_{com}, ϵ)-private *on parameters* par*, if for any t-time adversary* $\mathcal{A}$ *with at most* q_{com} *oracle accesses to procedures* $\mathsf{Com}(\mathsf{tk}_0)$ *and* $\mathsf{Com}(\mathsf{tk}_1)$*, we have* $|p_0 - p_1| \leq \epsilon$ *where*

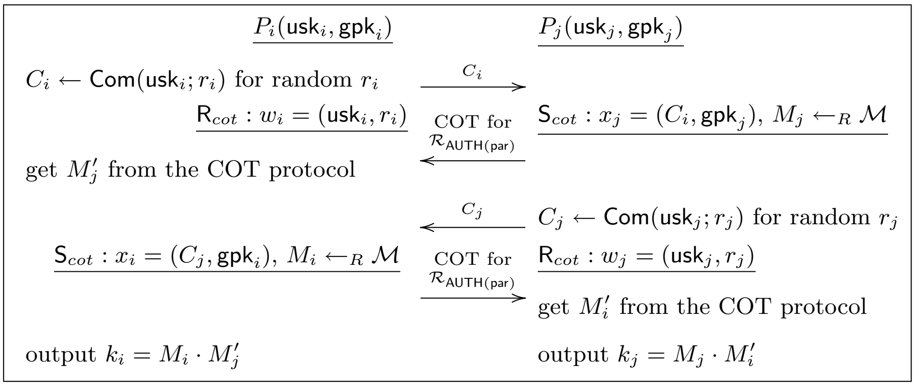
$$p_b \triangleq \Pr[\mathcal{A}^{\mathsf{Com}_{\mathsf{par}}(\mathsf{tk}_0), \mathsf{Com}_{\mathsf{par}}(\mathsf{tk}_1)}(\mathsf{par}, C_b) = 1 \mid \mathsf{tk}_0, \mathsf{tk}_1 \leftarrow_R \mathbb{U},\ C_b \leftarrow \mathsf{Com}_{\mathsf{par}}(\mathsf{tk}_b)]$$

where the probability additionally goes over the randomness of $\mathcal{A}$ *and* Com*.*

6.3 Construction of SH's from VLR-GS and Private COT

Assume we have a VLR-GS scheme consisting of procedures $\mathsf{Setup}_{\mathsf{GS}}$, KeyGen, Cert, Ver, Com, and $\mathsf{TraceCom}$ which satisfy all the above requirements. Assume also a private COT protocol for relation $\mathcal{R}_{\mathsf{AUTH(par)}}$ and message space $\mathcal{M}$ corresponding to this VLR-GS scheme. An SH scheme (Setup, KGen, $\mathsf{Handshake}$, Trace) is constructed as follows:

- Setup is the same as $\mathsf{Setup}_{\mathsf{GS}}$, and keyspace $\mathcal{K}$ is the message space $\mathcal{M}$.
- KGen, on input par, first computes $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})$, then computes $(\mathsf{tk}_i, \mathsf{scr}_i) \leftarrow \mathsf{Cert}_{\mathsf{par}}(\mathsf{sk})$ for $i = 1, .., N$, and outputs $(\mathsf{gpk}, \mathbf{usk}, \mathbf{urt})$ where $\mathsf{gpk} = \mathsf{pk}$, and for all i we assign $\mathbf{usk}[i] \leftarrow (\mathsf{tk}_i, \mathsf{scr}_i)$ and $\mathbf{urt}[i] \leftarrow \mathsf{tk}_i$.
- Protocol $\mathsf{Handshake}$, executed between players P_i and P_j: Player P_i runs the protocol on inputs $(\mathsf{usk}_i, \mathsf{gpk}_i)$ for some group in which P_i is a member. Similarly P_j runs it on $(\mathsf{usk}_j, \mathsf{gpk}_j)$ for some group in which P_j is a member. The protocol proceeds as in Figure 3.



$P_i(\mathsf{usk}_i, \mathsf{gpk}_i)$
$P_j(\mathsf{usk}_j, \mathsf{gpk}_j)$
$C_i \leftarrow \mathsf{Com}(\mathsf{usk}_i; r_i)$ for random r_i
$\xrightarrow{C_i}$
$\mathsf{R}_{cot} : w_i = (\mathsf{usk}_i, r_i)$
COT for $\mathcal{R}_{\mathsf{AUTH(par)}}$
$\mathsf{S}_{cot} : x_j = (C_i, \mathsf{gpk}_j)$, $M_j \leftarrow_R \mathcal{M}$
get M'_j from the COT protocol
$\xleftarrow{C_j}$
$C_j \leftarrow \mathsf{Com}(\mathsf{usk}_j; r_j)$ for random r_j
$\mathsf{S}_{cot} : x_i = (C_j, \mathsf{gpk}_i)$, $M_i \leftarrow_R \mathcal{M}$
COT for $\mathcal{R}_{\mathsf{AUTH(par)}}$
$\mathsf{R}_{cot} : w_j = (\mathsf{usk}_j, r_j)$
get M'_i from the COT protocol
output $k_i = M_i \cdot M'_j$
output $k_j = M_j \cdot M'_i$

Fig. 3. $\mathsf{Handshake}$ between P_i and P_j with inputs $(\mathsf{usk}_i, \mathsf{gpk}_i)$ and $(\mathsf{usk}_j, \mathsf{gpk}_j)$

- Trace, on inputs tr and tk, outputs $\mathsf{TraceCom}_{\mathsf{par}}(\mathsf{tr}, \mathsf{tk})$.

For lack of space we omit the proofs of the following theorems:

Theorem 3 (SH Security). *For any* par *outputted by* $\mathsf{Setup}_{\mathbf{GS}}$*, if* $\mathsf{Com}_{\mathsf{par}}$ *is traceable,* Π_{cert} *is* (t_1, q_1, ϵ_1)*-unforgeable on* par*, and if the COT protocol for* $\mathcal{R}_{\mathsf{auth(par)}}$ *is* $(t_2, t_{ext}, q_{ext}, d, e)$*-strongly-secure and sender-private with soundness error* δ*, and* (t_3, ϵ_3)*-receiver-private, then the above SH scheme is* $(t', q_{sh}, q_1, \epsilon')$*-secure, where* $t' = \min\{t_2, (t_1 - t_{ext})/(q_{ext} + 1), t_3\}$*,* $\epsilon' = (q_1 + 1)(\epsilon_1/d)^{1/e} + \delta) + q_1 \cdot q_{sh} \cdot \epsilon_3$.

Theorem 4 (SH Privacy). *For any* par *outputted by* $\mathsf{Setup}(\kappa)$*, if the* $\mathsf{Com}_{\mathsf{par}}$ *algorithm is* (t_1, q_1, ϵ_1)*-private and is uniquely traceable, if* P_{cert} *is* (t_2, q_2, ϵ_2)*-unforgeable and if the COT protocol for* $\mathcal{R}_{\mathsf{auth(par)}}$ *is* $(t_3, t_{ext}, q_{ext}, d, e)$*-strongly-secure-and-sender-private with soundness error* δ*,and* (t_4, ϵ_4)*-receiver-private,* *then the above SH scheme is* $(t', q_{sh}, q_2, \epsilon')$*-private, where* $t' = \min\{t_1, (t_2 - t_{ext})/(q_{ext}+1), t_3, t_4\}$*,* $\epsilon' = \epsilon_1 + ((q_2+1)(\epsilon_2/d)^{1/e} + \delta) + (q_2 \cdot q_{sh} \cdot \epsilon_4) + \epsilon_4$*, and* $q_{sh} \leq q_1$*.*

Note that the SH scheme presented above is a generic construction from appropriate VLR-GS components and an associated private COT protocol. For lack of space we omit from these proceedings a description of a concrete implementation where all components are instantiated with those used in the VLR-GS scheme of [BS04]. However, it is easy to see that relation $\mathcal{R}_{\mathsf{AUTH}}$ defined by these components can be transformed to a special case of relation $\mathcal{R}_{\mathsf{REP}(\Phi)}$ of Section 5, and therefore an efficient private COT protocol for this relation is implied by the private COT for $\mathcal{R}_{\mathsf{REP}(\Phi)}$ given in Figure 2.

Acknowledgments. The authors want to thank Anna Lysyanskaya and Yuval Ishai for stimulating discussions, and Gene Tsudik for frequent productive discussions and for a crucial suggestion which triggered this work, namely to look into converting some group signature into a "group-signature-based" envelope.

References

[AIR01] Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BBW06] Barth, A., Boneh, D., Waters, B.: Privacy in encrypted content distribution using private broadcast encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 52–64. Springer, Heidelberg (2006)

[BCK01] Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key-exchange protocols. In: Symposium on Theory of Computing (2001)

[BDS$^+$03] Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.-C.: Secret handshakes from pairing-based key agreements. In: IEEE Symposium on Security and Privacy, pp. 180–196 (2003)

[BGW05] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)

[BK04] Blake, I.F., Kolesnikov, V.: Strong conditional oblivious transfer and computing on intervals. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 515–529. Springer, Heidelberg (2004)

[Bou00] Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)

[BS04] Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security, pp. 168–177 (2004)

[BSW07] Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)

[CDS94] Cramer, R., Damgård, I., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

[CJT04] Castelluccia, C., Jarecki, S., Tsudik, G.: Secret handshakes from ca-oblivious encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)

[CL01] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

[CM99] Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)

[COR99] Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)

[Cre00] Di Crescenzo, G.: Private selective payment protocols. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 72–89. Springer, Heidelberg (2001)

[CS01] Cramer, R., Shoup, V.: Universal hash proofs and and a paradigm for adaptive chosen ciphertext secure public-key encryption. Electronic Colloquium on Computational Complexity (ECCC), 8(072) (2001)

[CvH91] Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

[Dam00] Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, p. 418. Springer, Heidelberg (2000)

[GPSW06] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)

[JKT07] Jarecki, S., Kim, J., Tsudik, G.: Group secret handshakes or affiliation-hiding authenticated group key agreement. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 287–308. Springer, Heidelberg (2006)

[JL07] Jarecki, S., Liu, X.: Unlinkable secret handshakes and key-private group key management schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)

[JL08] Jarecki, S., Liu, X.: Affiliation-hiding envelope and authentication schemes with efficient support for multiple credentials. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 715–726. Springer, Heidelberg (2008)

[KP97] Kilian, J., Petrank, E.: Identity escrow. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 169–185. Springer, Heidelberg (1997)

[LL07] Laur, S., Lipmaa, H.: A new protocol for conditional disclosure of secrets and its applications. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 207–225. Springer, Heidelberg (2007)

[MP03] Micciancio, D., Petrank, E.: Simulatable commitments and efficient concurrent zero-knowledge. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 140–159. Springer, Heidelberg (2003)

[NNL02] Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. Electronic Colloquium on Computational Complexity (ECCC), 043 (2002)

[Ped91] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)

[Yao86] Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)

Randomizable Proofs and Delegatable Anonymous Credentials

Mira Belenkiy[1], Jan Camenisch[2], Melissa Chase[3], Markulf Kohlweiss[4], Anna Lysyanskaya[5], and Hovav Shacham[6]

[1] Microsoft
mibelenk@microsoft.com
[2] IBM Zurich Research Laboratory
jca@zurich.ibm.com
[3] Microsoft Research
melissac@microsoft.com
[4] K.U. Leuven
mkohlwei@esat.kuleuven.be
[5] Brown University
anna@cs.brown.edu
[6] UC San Diego
hovav@cs.ucsd.edu

Abstract. We construct an efficient delegatable anonymous credentials system. Users can anonymously and unlinkably obtain credentials from any authority, delegate their credentials to other users, and prove possession of a credential L levels away from a given authority. The size of the proof (and time to compute it) is $O(Lk)$, where k is the security parameter. The only other construction of delegatable anonymous credentials (Chase and Lysyanskaya, Crypto 2006) relies on general non-interactive proofs for NP-complete languages of size $k\Omega(2^L)$. We revise the entire approach to constructing anonymous credentials and identify *randomizable* zero-knowledge proof of knowledge systems as the key building block. We formally define the notion of randomizable non-interactive zero-knowledge proofs, and give the first instance of *controlled rerandomization* of non-interactive zero-knowledge proofs by a *third-party*. Our construction uses Groth-Sahai proofs (Eurocrypt 2008).

1 Introduction

Access control is one of the most fundamental problems in security. We frequently need to answer the question: does the person requesting access to a resource possess the required credentials? A credential typically consists of a certification chain rooted at some authority responsible for managing access to the resource and ending at the public key of the user in question. The user presents the credential and demonstrates that he knows the corresponding secret key. Sometimes, the trusted authority issues certificates *directly* to each user (so the length of each certification chain is 1). More often, the authority *delegates* responsibility. A system administrator allows several webmasters to use his server. A webmaster can create several forums, with different moderators for each forum. Moderators approve some messages, reject others, and even give favored users unlimited posting privileges. Imagine the burden on the system administrator if he had to approve every single moderator and user for every single forum.

We want cryptographic credentials to follow the same delegation model as access control follows in the real world. The system administrator can use his public key to sign a webmaster's public key, creating a credential of length 1. In general, a user with a level L credential can sign another user's public key and give him his credential chain, to create a level $L + 1$ credential.

The design of an *anonymous* delegatable credential scheme in which participants can obtain, delegate, and demonstrate possession of credential chains without revealing any additional information about themselves is a natural and desirable goal. Our main contribution is the first efficient delegatable anonymous credential scheme. The only known construction of delegatable anonymous credentials, due to Chase and Lysyanskaya [CL06], needs $k^{\Omega(L)}$ space to store a certification chain of length L (for security parameter k), and therefore could not tolerate non-constant L. Our solution is *practical*: all operations on chains of length L need $\Theta(kL)$ time and space.

Pseudonymous systems. Prior work on anonymous credentials [CL02b, BCKL08] created systems where each user has one secret key but multiple "public keys." Given a secret key sk_A, Alice can create a new public key by choosing a random value *open* and publishing a commitment $pk_A = \mathsf{Commit}(sk_A, open)$. Alice could register pk_A with Oliver and pk'_A with Olga.

Oliver can give Alice a credential by signing the statement that the value in commitment pk_A has some attribute. Alice can then show pk'_A to Olga and prove that Oliver signed that the value in pk'_A had that attribute. This works because pk_A and pk'_A are commitments to the same value sk_A. The chief building block of an anonymous credential scheme is a signature scheme that lends itself to the design of efficient protocols for (1) obtaining a signature on a committed value; and (2) proving that a committed value has been signed.

Why delegation is a challenging problem. There is no straightforward transformation of anonymous credential schemes [Cha85, Bra99, LRSW99, CL01, BCKL08] into delegatable schemes. Suppose instead of giving Alice a direct credential, Oliver delegates his own credential to Alice. If Oliver gives Alice a *sig* on Oliver's secret key, then Alice could learn who gave Oliver his credential. Generalizing this approach would reveal to Alice the identity of every person in Oliver's credential chain.

Our approach. Instead of giving Alice his signature, Oliver gives Alice a non-interactive proof-of-knowledge of the signature. We show how Alice can (1) delegate the credential by extending the proof and (2) rerandomize the proof every time she shows (or extends it) to preserve her anonymity.

Let's say Oliver is a credential authority and Alice wants to obtain the credential directly from Oliver (so her certification chain will be of length 1). Under the old approach, they would run a secure two-party protocol as a result of which Alice obtains a signature $\sigma_{pk_O}(sk_A)$ on sk_A, while Oliver gets no output. Under the new approach, Alice's output is (C_A, π_A), where C_A is a commitment to her secret key sk_A, and π_A is a *proof of knowledge* of Oliver authenticating the contents of C_A. Note that a *symmetric* authentication scheme is sufficient because *no one ever sees the authenticator*; all verification is done on the proof of knowledge. The symmetric key sk_O is still known only to Oliver; we create a "public" key C_O that is simply a commitment to sk_O.

How can Alice use this credential anonymously? If the underlying proof system is *malleable* in just the right way, then given (C_A, π_A) and the opening to C_A, Alice can compute (C'_A, π'_A) such that C'_A is another commitment to her sk_A that she can successfully open, while π'_A is a proof of knowledge of Oliver authenticating the contents of C'_A. Malleability is usually considered a bug rather than a feature. However, in combination with the correct extraction properties, we still manage to guarantee that these randomizable proofs give us a useful building block for the construction.

How does Alice delegate her credential to Bob? Alice and Bob can run a secure protocol as a result of which Bob obtains (C_B, π_B) where C_B is a commitment to Bob's secret key sk_B and π_B is a proof of knowledge of an authenticator issued by the owner of C'_A on the contents of C_B. Now, essentially, the set of values $(C'_A, C_B, \pi'_A, \pi_B)$ together indicate that the owner of C'_A got a credential from Oliver and delegated to the owner of C_B, and so it constitutes a proof of possession of a certification chain. Moreover, it hides the identity of the delegator Alice! Now Bob can, in turn, use the randomization properties of the underlying proof system to randomize this set of values so that it becomes unlinkable to his original pseudonym C_B; he can also, in turn, delegate to Carol.

Randomizable proof systems. The key to our construction is a randomizable proof system that lets the prover (1) randomize the proof *without knowing the witness*, and (2) *control* the outcome of the randomization process. This is a fundamentally new notion, and one we think will be of independent interest. We give a formal definition and show that we can instantiate it by adding a randomization procedure to the pairing-based proof system of Groth and Sahai [GS08]. Our use of pairings is not merely a matter of efficiency – we do not know of any proof system based on general assumptions that can be randomized.

In fact the Groth-Sahai proofs allow us to go beyond merely randomizing the proof to actually change the statements we are proving. What do we mean by this? Groth-Sahai proofs make statements about the values inside commitments. Let $C = \mathsf{Commit}(x, \mathit{open})$. A prover who knows (x, open) can choose a new value open' so that in the rerandomized proof, C is transformed to $C' = \mathsf{Commit}(x, \mathit{open}')$. Otherwise, the prover can choose whether to leave C unchanged or randomize it to C' that uses a some random open' unknown to the prover. This fine level of control together with the basic randomization property gives a very useful building block, which is crucial in our application.

There has been prior work on some related notions: Burmester et al [BDI+99] show a third party can help randomize proofs during the execution of an *interactive* protocol to prevent subliminal channels. De Santis and Yung [DSY90] propose the notion of meta proofs, in which anyone who holds a proof for a given statement can generate a proof that there exists a proof for the statement. Neither of these approaches work for our scenario because we need to randomize *non-interactive* proofs, and, unlike a meta-proof, the randomized proof must be indistinguishable from the original.

Our delegatable credentials construction. We construct delegatable credentials using randomizable proofs. By concatenating rerandomized credential chains, we can create a credential chain of length L that takes $O(L)$ space. Our strong anonymity properties are an immediate consequence of rerandomization: each showing of the credential is unlinkable and users do not learn the identities of delegators in their own credential chain.

Our solution (1) prevents adversarial users from mixing and matching pieces of different credential chains to create unauthorized credential chains and (2) protects the user's anonymity even when the adversary delegates *to* the user. We solve the second problem by creating an authentication scheme (symmetric signature scheme) that is secure even when the adversary gets a signature on the user's secret key.

Attributes. Our delegatable anonymous credentials system lets users add human-readable attributes to each credential. Oliver can give Alice a level 1 credential with attribute "webmaster of Crypto Forum". Alice can then delegate her credential to Bob with attribute "moderator of Crypto Forum". As a result, Bob can log on to the server anonymously and prove that the "webmaster of Crypto Forum" made him the "moderator of Crypto Forum". Our construction lets users add as many attributes as they want to each credential, allowing for the expressibility that we see in modern (non-anonymous) access control systems.

Advanced abuse prevention mechanisms. Our construction shows how to efficiently implement maximum anonymity at all levels and all roles in the delegation chain—with the exception of the credential authority. Some applications will not require this full anonymity. Indeed, a large number of abuse prevention mechanisms for anonymous credentials (anonymity revocation [CL01], credential revocation [CL02a], limited show [CHK$^+$06]) aim at striking a balance between privacy and accountability. Concerning our own scheme we make three simple observations: (i) global traceability can be achieved by providing a trusted tracing authority with the extraction trapdoors for the common parameters of the Groth Sahai (GS) proof system; (ii) at the last level of the delegation chain, we can make use of all abuse prevention mechanisms known for traditional anonymous credentials; (iii) many abuse prevention mechanisms known from the literature can be adapted to our construction by replacing traditional sigma proofs [Dam02] with GS proofs [GS08].

Other related work. At first glance, our delegatable credentials scenario might resemble the HIBE or HIBS settings [GS02, BBG05], where a root delegator can issue decryption or signing keys to recipients, who in turn can delegate subkeys to lower level participants. There are two key differences between such a

HIBE/HIBS scheme and anonymous credential schemes: (1) In HIBE/HIBS two users with the same attributes are completely interchangeable while an anonymous credentials system gives them distinct sets of pseudonyms and (2) anonymous credentials allow a user to show that he has obtained valid credentials from two independent authorities.

In a somewhat different direction, Barak [Bar01] presented a general (inefficient) construction for delegatable signatures. In that work, the goal was a signature scheme which would allow the signer to delegate signing rights for a restricted message space, such that signatures generated with the delegated key are indistinguishable from the originals. In our setting, we want the opposite: once we delegate a credential, the delegatee should be able to issue lower level credentials to any users he chooses, however we require that credentials at different levels be clearly distinguishable. Finally, the definition in [Bar01] does not consider anonymity between the delegator and the delegatee, while we do.

Our contribution and organization of the paper. We (1) define and construct a randomizable NIZKPK (Section 2) and (2) define and construct an efficient delegatable anonymous credential system (Section 3). We also create an appropriate message authentication scheme and some other additional building blocks for the delegatable credentials scheme. We show how these building blocks can be instantiated under appropriate assumptions about groups with bilinear maps.

2 Randomizable NIZK Proof Systems

Let $R(params, y, w)$ be any polynomial-time computable relation. A non-interactive proof system for relation R allows the prover to convince a verifier that for some instance y there exists a witness w such that $R(params, y, w)$, where $params$ is a common (public) reference string. The prover generates a proof $\pi \leftarrow \mathsf{Prove}(params, y, w)$, the verifier checks it via $\mathsf{VerifyProof}(params, y, \pi)$. A trusted third party runs $params \leftarrow \mathsf{Setup}(1^k)$ once to initialize the system.

Informally, zero-knowledge captures the notion that a verifier learns nothing from the proof but the truth of the statement. Witness indistinguishability merely guarantees that the verifier learns nothing about which witness was used in the proof. Soundness means an adversary cannot convince an honest verifier of a false statement. Completeness means all honest verifiers accept all correctly computed proofs. See [GMR89, Gol00, BFM88, FLS99] for formal definitions.

We define randomizable proof systems, which have an additional algorithm $\mathsf{RandProof}$ that takes as input a proof π for instance y in relation R, and produces a new proof for the same statement y. The resulting proof must be indistinguishable from a new proof for y. We allow the adversary to choose the instance y, the proof π that is used as input for $\mathsf{RandProof}$, and the witness w that is used to form a new proof of the same statement. Formally:

Definition 1. *We say that* $\mathsf{Setup}, \mathsf{Prove}, \mathsf{VerifyProof}, \mathsf{RandProof}$ *constitute a randomizable proof system if the following property holds. For all ppt.* $(\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* ν *such that:*

$$\begin{aligned}&Pr[params \leftarrow \mathsf{Setup}(1^k); (y, w, \pi, state) \leftarrow \mathcal{A}_1(params);\\ &\quad \pi_0 \leftarrow \mathsf{Prove}(params, y, w); \pi_1 \leftarrow \mathsf{RandProof}(params, y, \pi);\\ &\quad b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}_2(state, \pi_b) :\\ &\quad R_L(y, w) \wedge \mathsf{VerifyProof}(params, y, \pi) = 1 \wedge b = b'] \leq 1/2 + \nu(k)\ .\end{aligned}$$

2.1 Instantiating a Randomizable Proof System

Randomization is a fundamentally new property. It is not clear how one might randomize proofs in any of the existing NIZK proof systems [BDMP91, KP98, FLS99] *without knowing the witness.* The one exception is the recent proof system of Groth and Sahai [GS08] (an extension of [GOS06]), which gives witness indistinghishable (and in some cases zero-knowledge) NIPKs. We will show how to add a randomization procedure to Groth-Sahai proofs.

Summary of Groth-Sahai proofs. Let $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$ be the setup for pairing groups of prime order p, with pairing $e : G_1 \times G_2 \to G_T$, and g, h generators of G_1, G_2 respectively.[1]

The instance consists of the coefficients of a pairing product equation:

$\{a_q\}_{q=1\ldots Q} \in G_1$, $\{b_q\}_{q=1\ldots Q} \in G_2$, $t \in G_T$, and $\{\alpha_{q,m}\}_{q=1\ldots Q, m=1\ldots M}$, $\{\beta_{q,n}\}_{q=1\ldots Q, n=1\ldots N} \in Z_p$. The prover knows $\{x_m\}_{m=1}^{M}$, $\{y_n\}_{n=1}^{N}$ that satisfy the pairing product equation $\prod_{q=1}^{Q} e(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t$.

The prover creates perfectly binding, computationally hiding commitments $\{c_m\}_{m=1\ldots M}$ and $\{d_n\}_{n=1\ldots N}$ for all values x_m, y_n in G_1 and G_2 respectively. The instance is the pairing product equation (e.g. its coefficients) and the above commitments, while the witness, known only to the prover, is the values *and openings* of these commitments.

We now describe how to construct the proof. Let M_1, M_2, and M_T be R-modules for some ring R, and let $E\colon M_1 \times M_2 \to M_T$ be a bilinear map. Also let μ_1, μ_2, μ_T be efficiently computable embeddings that map elements of G_1, G_2, G_T into M_1, M_2, M_T, respectively. The public parameters $params_{PK}$ contain elements $u_1, \ldots, u_I \in M_1$, $v_1, \ldots, v_J \in M_2$ and values $\eta_{h,i,j}$, $1 \leq i \leq I$, $1 \leq j \leq J$, and $1 \leq h \leq H$.

To create a Groth-Sahai commitment to $x \in G_1$, choose random opening $open = (r_1, \ldots, r_I) \leftarrow R^I$, and compute $c = \mu_1(x) \cdot \prod_{i=1}^{I} u_i^{r_i}$. Elements $y \in G_2$ are committed to in the same way using μ_2 and $v_1, \ldots, v_J \in M_2$, and an opening vector $open \in R^J$. For simplicity we assume that $\mathsf{GSCommit}(params_{PK}, m, open)$ first determines whether $m \in G_1$ or $m \in G_2$ and then follows the appropriate instructions.

Groth and Sahai [GS08] show how to efficiently compute proofs $\{\pi_i\}_{i=1}^{I}$, $\{\psi_j\}_{j=1}^{J}$ that prove that the openings of the c_m and d_n satisfy a pairing product equation. The verifier computes, for all $1 \leq q \leq Q$, $\hat{c}_q \leftarrow \mu_1(a_q) \cdot \prod_{m=1}^{M} c_m^{\alpha_{q,m}}$

[1] For simplicity, we do not consider Groth-Sahai proofs for composite order groups.

and $\hat{d}_q \leftarrow \mu_2(b_q) \cdot \prod_{n=1}^{N} d_n^{\beta_{q,n}}$. Then the verifier checks that $\prod_{q=1}^{Q} E(\hat{c}_q, \hat{d}_q) = \mu_T(t) \cdot \prod_{i=1}^{I} E(u_i, \pi_i) \cdot \prod_{j=1}^{J} E(\psi_j, v_j)$.

Randomizing Groth-Sahai proofs. RandProof gets as input an instance with the $a_q, b_q, t, \alpha_{q,m}, \beta_{q,n}$ values as well as the proof $[(\pi_1, \ldots, \pi_I, \psi_1, \ldots, \psi_J), \Pi]$. Π contains the internal commitments $c_1, \ldots, c_M$ and $d_1, \ldots, d_N$.

The algorithm first chooses randomization exponents $(s_{1,1}, \ldots, s_{M,I})$ and $(z_{1,1}, \ldots, z_{N,J})$ at random from Z_p. It then rerandomizes the commitments c_m and d_n to $c'_m = c_m \cdot \prod_{i=1}^{I} u_i^{s_{m,i}}$ and $d'_n = d_n \cdot \prod_{j=1}^{J} v_j^{z_{n,j}}$. Then it computes $\hat{s}_{q,i} = \sum_{m=1}^{M} s_{m,i} \cdot \alpha_{q,m}$, $\hat{z}_{q,j} = \sum_{n=1}^{N} z_{n,j} \cdot \beta_{q,n}$, and $D'_q \leftarrow \mu_2(b_q) \cdot \prod_{n=1}^{N} d_n'^{\beta_{q,n}}$ and $C_q \leftarrow \mu_1(a_q) \cdot \prod_{m=1}^{M} c_m^{\alpha_{q,m}}$. Next, the prover sets $\pi'_i \leftarrow \pi_i \cdot \prod_{q=1}^{Q} (D'_q)^{\hat{s}_{q,i}}$ and $\psi'_j \leftarrow \psi_j \cdot \prod_{q=1}^{Q} (C_q)^{\hat{z}_{q,j}}$. These π'_i and ψ'_j will satisfy the verification equation for the new commitments.

Now the prover must make a certain technical step to fully randomize the proof. Intuitively, for every set of commitments, there are many proofs $(\pi_1, \ldots, \pi_I, \psi_1, \ldots, \psi_J)$ that can satisfy the verification equation. Given one such proof, we can randomly choose another: The prover chooses $t_{i,j}, t_h \leftarrow R$, and multiplies each $\pi'_i := \pi'_i \cdot \prod_{j=1}^{J} v_j^{t_{i,j}}$ and each $\psi'_j := \psi'_j \cdot \prod_{i=1}^{I} u_i^{\sum_{h=1}^{H} t_h \eta_{h,i,j}} \prod_{i=1}^{I} u_i^{t_{i,j}}$. See [GS08] for a detailed explanation.

The algorithm outputs the new proof $[(\pi'_1, \ldots, \pi'_I, \psi'_1, \ldots, \psi'_J), \Pi']$ where Π' contains the internal commitments $c'_1, \ldots, c'_M$ and $d'_1, \ldots, d'_N$. See full version [BCC$^+$08] for details. A similar approach works for composite order groups.

Composable Proofs. Groth-Sahai proofs are *composable* witness indistinguishable, and in some cases composable zero-knowledge. To simplify our definitions and proofs, we use a similar notion for randomizability.

In a composable (under the definition of Groth and Sahai [GS08]) non-interactive proof system there exists an algorithm SimSetup that outputs *params* together with a trapdoor *sim*, such that *params* output by SimSetup is indistinguishable from those output by Setup. Composable witness-indistinguishability (or zero-knowledge) requires that, under these parameters, the witness-indistinguishability (resp. zero-knowledge) property holds *even when the adversary is given the trapdoor sim*. Groth-Sahai commitments are perfectly hiding under the simulated parameters. (Under the honest parameters they are perfectly binding.) In the same spirit, we say the *composable* randomizability property must hold even when the distinguisher is given the trapdoor *sim*.

2.2 Malleable Proofs and Randomizable Commitments

For our application, randomizing proofs is not sufficient. We also need to randomize (anonymize) the statement that we are proving. Consider a family of transformations $\{Y_s, P_s\}_{s \in S}$ that transform the instance and the proof respectively (for us, S is the set of all possible commitment openings). We require that $\forall (y, \pi), \forall s \in S$, if π is a valid proof for y, then $P_s(\pi)$ is a valid proof for $Y_s(y)$.

Definition 2. *We say that* Setup*,* Prove*,* $\mathsf{VerifyProof}$*,* $\mathsf{RandProof}$*,* $\{Y_s, P_s\}_{s\in S}$*, constitute a* Y*-malleable randomizable proof system, if for all ppt.* $\mathcal{A}$ *there exists a negligible* ν *such that:*

$$Pr[params \leftarrow \mathsf{Setup}(1^k); (y, \pi, s) \leftarrow \mathcal{A}(params) : \\ \mathsf{VerifyProof}(params, y, \pi){=}1 \wedge \mathsf{VerifyProof}(params, Y_s(y), P_s(\pi)) = 0]{=}\nu(k) .$$

If we apply $\mathsf{RandProof}$ to $P_s(\pi)$, then the result will be indistinguishable from a random fresh proof for $Y_s(y)$.

Groth-Sahai proofs can be used to prove that *the values in a given set of commitments* form a solution to a specific set of pairing product equations; the commitments can be part of the proof or the instance y. In our application, we will need to anonymize not only the proof, but also the commitments in the instance.

Suppose a prover wants to show that some $\mathsf{Condition}$ holds for the values inside commitments $C_1, \ldots, C_n$. Then the instance is $y = (\mathsf{Condition}, C_1, \ldots, C_n)$, and the witness is $w = (x_1, open_1, \ldots, x_n, open_n, z)$, where $(x_i, open_i)$ is the opening of commitment C_i, while z is some value that has nothing to do with the commitments. We define the relation $R = \{(params, y, w) | C_1 = \mathsf{Commit}(params, x_1, open_1) \wedge \ldots \wedge C_n = \mathsf{Commit}(params, x_n, open_n) \wedge \mathsf{Condition}(params, x_1, \ldots, x_n, z)\}$. A proof system supports randomizable commitments if there exist efficient algorithms Y and P, such that on input (s, y, π), where $s = (open'_1, \ldots, open'_n)$ and $\pi \leftarrow \mathsf{Prove}(params, y, w)$, (1) $Y(s, y)$ outputs instance $y' = (\mathsf{Condition}, C'_1, \ldots, C'_n)$, where $C'_i = Commit(params, x_i, open_i + open'_i)$, (2) $P(s, \pi)$ outputs a proof π' for instance y', and (3) Y and P fulfill the malleability requirements of Definition 2.

Lemma 1. *The Groth-Sahai proof system is malleable with respect to the randomness in the commitments.* See full version [BCC+08] for details.

Remark 1. To simplify notation, $\mathsf{RandProof}$ will take $s = (open'_1, \ldots, open'_n)$ as input, apply P_s, and then run the randomization algorithm. To leave C_i unchanged, we set $open'_i = 0$.

2.3 Partially Extractable Non-interactive Proofs of Knowledge

A NIPK system is a non-interactive proof system that is *extractable.* We recall the notion of *f-extractability* [BCKL08], which is an extension of the original definition of extractability [SCP00]. In an extractable proof system, there exists a ppt. extractor $(\mathsf{PKExtractSetup}, \mathsf{PKExtract})$. $\mathsf{PKExtractSetup}(1^k)$ outputs $(td, params)$ where $params$ is distributed identically to the output of $\mathsf{Setup}(1^k)$. For all polynomial time adversaries $\mathcal{A}$, the probability that $\mathcal{A}(1^k, params)$ outputs (y, π) such that $\mathsf{VerifyProof}(params, y, \pi) = \mathsf{accept}$ and $\mathsf{PKExtract}(td, y, \pi)$ fails to extract a witness w such that $R(params, y, w) = \mathsf{accept}$ is negligible in k. We have *perfect* extractability if this probability is 0. f-Extractability means that the extractor $\mathsf{PKExtract}$ only has to output a w' such that $\exists w$:

$R(params, y, w) = \mathsf{accept} \wedge w' = f(params, w)$. If $f(params, \cdot)$ is the identity function, we get the usual notion of extractability.

Let C be an unconditionally binding commitment. By 'x in C' we mean $\exists open : C = \mathsf{Commit}(params, x, open)$. We use NIPK notation [CS97, BCKL08], to denote an f-extractable NIPK for instance $(C_1, \ldots, C_n, \mathsf{Condition})$ with witness $(x_1, open_1, \ldots, x_n, open_n, z)$:

$$\pi \leftarrow \mathsf{NIPK}[x_1 \text{ in } C_1, \ldots, x_n \text{ in } C_n]\{(\ f(params,\ (x_1, open_1, \ldots, x_n, open_n, z)\ )\ ) : \mathsf{Condition}(params, x_1, \ldots, x_n, z)\}.$$

The f-extractability property ensures that if $\mathsf{VerifyProof}$ accepts then we can extract $f(params, (x_1, open_1, \ldots, x_n, open_n, z))$ from π, such that x_i is the content of the commitment C_i, and $\mathsf{Condition}(params, x_1, \ldots, x_n, z)$ is satisfied.

In our notation, $\pi \in \mathsf{NIPK}[\ldots\}$ means that $\mathsf{VerifyProof}$ accepts the proof π for instance $(C_1, \ldots, C_n, \mathsf{Condition})$. To further abbreviate notation, we omit *params* and assume that $\mathsf{Condition}$ is clear from the context, and so the sole inputs to $\mathsf{VerifyProof}$ are $(C_1, \ldots, C_n)$ and π. If the proof is zero-knowledge instead of merely witness indistinguishable, we will write NIZKPK.

The *concatenation* of two proofs π and π' is a proof $\pi \circ \pi'$ that combines all the commitments and proves the AND of the two conditions. If a proof π proves a condition about a set of commitments $\mathcal{C}$, a *projection* $\pi' = \pi \circ \mathcal{S}$ proves a condition about the contents of the subset $\mathcal{C} \setminus \mathcal{S}$ of commitments. A projected proof π' is obtained by removing the commitments in $\mathcal{S}$ from the instance and appending them to the proof.

Groth-Sahai proofs give us NIPK of the form:

$$\mathsf{NIPK}_{\mathsf{GS}}[\{x_m \text{ in } c_m\}_{m=1}^{M}, \{y_n \text{ in } d_n\}_{n=1}^{N}]\{(x_1, \ldots, x_M, y_1, \ldots, y_N) : \prod_{q=1}^{Q} e(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t\}.$$

3 Delegatable Anonymous Credentials

An anonymous delegatable credential system has only one type of participant: users. Each user has a single secret key and uses it to generate different pseudonyms. User A with secret key sk_A can be known to user O as $Nym_A^{(O)}$ and to user B as $Nym_A^{(B)}$. Any user O can become an originator of a credential; all he needs to do is publish one of his pseudonyms Nym_O as his public key. If authority O issues user A a credential for $Nym_A^{(O)}$, then user A can prove to user B that $Nym_A^{(B)}$ has a credential from authority O. Credentials received directly from the authority are level 1 credentials, credentials that have been delegated once are level 2 credentials, etc. A delegatable credential system consists of the following algorithms:

$\mathsf{Setup}(1^k)$ outputs the trusted public parameters of the system, params_{DC}.

$\mathsf{Keygen}(\mathit{params}_{DC})$ creates the secret key of a party in the system.

$\mathsf{Nymgen}(\mathit{params}_{DC}, sk)$. On each run, the algorithm outputs a new pseudonym Nym with auxiliary info $\mathit{aux}(\mathit{Nym})$ for secret key sk.[2]

$\mathsf{Issue}(\mathit{params}_{DC}, \mathit{Nym}_O, sk_I, \mathit{Nym}_I, \mathit{aux}(\mathit{Nym}_I), \mathit{cred}, \mathit{Nym}_U, L)$ $\leftrightarrow \mathsf{Obtain}(\mathit{params}_{DC}, \mathit{Nym}_O, sk_U, \mathit{Nym}_U, \mathit{aux}(\mathit{Nym}_U), \mathit{Nym}_I, L)$ are the interactive algorithms that let a user I issue a level $L+1$ credential to a user U. The pseudonym Nym_O is the authority's public key, sk_I is the issuer's secret key, Nym_I is the issuer's pseudonym with auxiliary information $\mathit{aux}(\mathit{Nym}_I)$, cred is the issuer's level L credential rooted at Nym_O, sk_U is the user's secret key, and Nym_U is the user's pseudonym with auxiliary information $\mathit{aux}(\mathit{Nym}_U)$. If $L = 0$ then $\mathit{cred} = \epsilon$. The issuer gets no output, and the user gets a credential cred_U.

$\mathsf{CredProve}(\mathit{params}_{DC}, \mathit{Nym}_O, \mathit{cred}, sk, \mathit{Nym}, \mathit{aux}(\mathit{Nym}), L)$. Takes as input a level L credential cred from authority Nym_O, outputs a value $\mathit{credproof}$.

$\mathsf{CredVerify}(\mathit{params}_{DC}, \mathit{Nym}_O, \mathit{credproof}, \mathit{Nym}, L)$. Outputs accept if $\mathit{credproof}$ is a valid proof that the owner of pseudonym Nym possesses a level L credential with root Nym_O and reject otherwise.

3.1 Security Definition of Delegatable Credentials

We formally define a secure delegatable credential system in the full version [BCC+08]. Intuitively, the algorithms Setup, Keygen, Nymgen, VerifyAux, Issue, Obtain, CredProve, and CredVerify constitute a secure anonymous delegatable credential scheme if the following properties hold:

Correctness. We say that a credential *cred* is a proper credential, if for all of the user's pseudonyms, CredProve always creates a proof that CredVerify accepts.The delegatable credential system is correct if an honest user and an honest issuer can run Obtain $\leftrightarrow$ Issue and the honest user gets a proper credential.

Anonymity. The adversary's interactions with the honest parties in the real game should be indistinguishable from some ideal game in which pseudonyms, credentials and proofs are independent of the user's identity and delegation chain. The adversary should not even recognize a credential he delegated.

There must exist a simulator (SimSetup, SimProve, SimObtain, SimIssue). SimSetup produces parameters indistinguishable from those output by Setup, along with some simulation trapdoor *sim*. Under these parameters, we require that the following properties hold when even the adversary is given *sim*:

– Nym is distributed independently of sk.
– No adversary can tell if it is interacting with Issue run by an honest party with a proper credential, or with SimIssue which is not given the credential and the issuer's secret key, but only the name of the authority, the length of the credential chain, and the pseudonyms of the issuer and user.

[2] We do not address how to prove ownership of a pseudonym; in our constructions this involves interactively proving knowledge of the opening of a commitment.

– No adversary can tell if it is interacting with Obtain run by an honest party with secret *sk*, or with SimObtain that is only given the authority, the length of the credential chain, and the pseudonyms of the issuer and user.
– The simulator SimProve can output a fake *credproof* that cannot be distinguished from a real credential, even when SimProve is only told the authority, the length of the credential chain, and the pseudonym of the user.

Remark 2. Our definition implies the more complex but weaker definition in which the adversary only controls the public inputs to the algorithm. Our definition is easier to work with as we need only consider one protocol at a time, and only a single execution of each protocol.

Unforgeability. Each credential defines a specific delegation chain. We cannot monitor delegation between adversarial parties. However, we require that whenever the delegation chain shows that an honest player delegated a level L credential to some user, that delegation actually occurred.

In this game, all of the honest parties are controlled by a single oracle that keeps track of all honestly issued credentials. An adversary given access to this oracle should have only negligible probability of outputting a forged credential.

Let F be an efficiently computable bijection and a one-way function. There exists a ppt. algorithms ExtSetup and Extract with five properties:

– ExtSetup and Setup output identically distributed *params*.
– Under these parameters, pseudonyms are perfectly binding for *sk*.
– Extract always extracts the correct chain of L identities from an honestly generated level L *credproof*.
– Given an adversarially generated level L credential proof *credproof* from authority Nym_O for the pseudonym Nym, Extract will always produce either the special symbol $\perp$ or $f_0, \ldots f_L$ such that Nym_O is a pseudonym for $F^{-1}(f_0)$ and Nym is a pseudonym for $F^{-1}(f_L)$.
– No adversary can output a valid credential proof from which an unauthorized chain of identities is extracted. More formally we require that for all ppt. $\mathcal{A}$ there exists a negligible ν such that:

$$\begin{aligned}
&\Pr[(\mathit{params}_{DC}, \mathit{td}) \leftarrow \mathsf{ExtSetup}(1^k);\\
&\quad(\mathit{credproof}, \mathit{Nym}, \mathit{Nym}_O, L), \leftarrow \mathcal{A}^{\mathcal{O}(\mathit{params}_{DC},\cdot,\cdot)}(\mathit{params}_{DC}, \mathit{td});\\
&\quad(f_0, \ldots, f_L) \leftarrow \mathsf{Extract}(\mathit{params}_{DC}, \mathit{td}, \mathit{credproof}, \mathit{Nym}, \mathit{Nym}_O, L) :\\
&\quad\mathsf{CredVerify}(\mathit{params}_{DC}, \mathit{Nym}_O, \mathit{credproof}, \mathit{Nym}, L) = \mathsf{accept} \wedge\\
&\quad(\exists i \text{ such that } (f_0, i, f_{i-1}, f_i) \notin \mathsf{ValidCredentialChains} \wedge\\
&\quad f_{i-1} \in \mathsf{HonestUsers})] \leq \nu(k)\ ,
\end{aligned}$$

where $\mathcal{O}(\mathit{params}_{DC}, \mathit{command}, \mathit{input})$ describes all possible ways for the adversary $\mathcal{A}$ to interact with the delegatable credentials system: $\mathcal{A}$ can ask the oracle to add new honest users; the oracle generates $\mathit{sk} \leftarrow \mathsf{Keygen}(\mathit{params}_{DC})$, stores it in the list HonestUsers, and returns $F(\mathit{sk})$ as the handle. $\mathcal{A}$ can ask for new pseudonyms for existing honest users, referenced by $F(\mathit{sk})$, and he can provide a credential and ask an honest user to generate the corresponding proof. Finally, he can run the $\mathsf{Issue} \leftrightarrow \mathsf{Obtain}$ protocols on credentials of his choice, either between honest users, or with an adversarial issuer or obtainer. In this case, we need to keep track of which credentials are being issued, so that we will be able to identify a forgery. To do this, we use the $\mathsf{Extract}$ algorithm to extract the chain of identities behind each credential being issued and store it on the list $\mathsf{ValidCredentialChains}$. For details, see full version [BCC$^+$08].

Remark 3. We let the adversary track honest users' credentials and pseudonyms (but, of course, not their secret keys). Our definition is strictly stronger than one that uses a general oracle that does not reveal the credentials of honest users to the adversary. This approach results in a simpler definition and analysis.

3.2 Construction of Delegatable Credentials

We construct delegatable credentials using a randomizable NIZK proof system with randomizable commitments (as described in Section 2) and a message authentication scheme for a vector of messages $\boldsymbol{m}$ (in our basic scheme $|\boldsymbol{m}| = 2$)in the common parameters model: $\mathsf{AuthSetup}(1^k)$ outputs common parameters $params_A$, $\mathsf{AuthKg}(params_A)$ outputs a secret key sk, $\mathsf{Auth}(params_A, sk, \boldsymbol{m})$ outputs an authentication tag $auth$ that authenticates a vector of messages $\boldsymbol{m}$, and $\mathsf{VerifyAuth}(params_A, sk, \boldsymbol{m}, auth)$ accepts if $auth$ is a proper authenticator for $\boldsymbol{m}$ under key sk. (We will discuss the properties we will require from this authentication scheme after we present our delegatable credentials construction.)

The parameters of the delegatable credentials system combine the parameters $params_A$ from the authentication scheme and $params_{PK}$ from the composable and randomizable NIZKPK system and its associated commitment scheme Commit. We assume that all algorithms are aware of these parameters and omit them when appropriate to simplify our notation.

Intuition behind our construction. The keyspace of the authenticator must be a subset of the input space of the commitment scheme. Each user U has a secret key $sk_U \leftarrow \mathsf{AuthKg}(params_A)$, and forms his pseudonyms using Commit: $Nym_U = \mathsf{Commit}(sk_U, open_U)$. U can create arbitrarily many different pseudonyms by choosing new random values $open_U$. A user can act as an authority (originator) for credentials by making his pseudonym Nym_O publicly available.

The user's secret credential *cred* is a NIZKPK of a statement about U's specific *secret* pseudonym $S_U = \mathsf{Commit}(sk_U, 0)$ (this specific pseudonym does not in fact hide sk_U since it is formed as a deterministic function of sk_U). To show or delegate the credential, the user randomizes and mauls *cred* to obtain *credproof* using the $\mathsf{RandProof}$ algorithm described in Section 2. The resulting *credproof* is a proof about a proper pseudonym, $Nym_U = \mathsf{Commit}(sk_U, open)$ for a randomly chosen *open*.

Suppose a user with secret key sk_U has a level L credential from some authority O, and let $(sk_O, sk_1, \ldots, sk_{L-1}, sk_U)$ be the keys such that the owner of sk_i delegated the credential to sk_{i+1} (we let $sk_0 = sk_O$ and $sk_L = sk_U$). A *certification chain* is a list of authenticators $auth_1, \ldots, auth_L$, such that sk_i was used to generate authenticator $auth_{i+1}$ on message sk_{i+1}.

To make sure that pieces of different certification chains cannot be mixed and matched, we add a label r_i to each authenticator. The labels have to be unique for each authority and delegation level. Let H be a collision resistant hash function with an appropriate range. For a credential chain rooted at Nym_O, we set $r_i = H(Nym_O, i)$. Each $auth_i$ is then an output of $\mathsf{Auth}(params_A, sk_{i-1}, (sk_i, r_i))$. Let F be an efficiently computable bijection. The user U's level L private credential *cred* is a proof of the form

$$\mathsf{NIZKPK}[sk_0 \text{ in } Nym_O; sk_L \text{ in } S_U]\{(F(sk_0), \ldots, F(sk_L), auth_1, \ldots, auth_L) : \\ \mathsf{VerifyAuth}(sk_0, (sk_1, r_1), auth_1) \wedge \ldots \wedge \mathsf{VerifyAuth}(sk_{L-1}, (sk_L, r_L), auth_L)\}$$

Full construction. Let PKSetup, PKProve, PKVerify, and RandProof be a randomizable NIPK system and let AuthSetup, AuthKg, Auth, VerifyAuth be an authentication scheme, and let $H : \{0,1\}^* \rightarrow Z_p$ be a hash function.

$\mathsf{Setup}(1^k)$. Use $\mathsf{AuthSetup}(1^k)$ to generate $params_A$ and $\mathsf{PKSetup}(1^k)$ to generate $params_{PK}$; choose the hash function H (as explained above); and output $params_{DC} = (params_A, params_{PK}, H)$.

$\mathsf{Keygen}(params_{DC})$. Run $\mathsf{AuthKg}(params_A)$ and output the secret key sk.

$\mathsf{Nymgen}(params_{DC}, sk)$. Choose random *open*, compute $Nym = \mathsf{Commit}(params_{PK}, sk, open)$ and output pseudonym Nym and auxiliary information *open*.

$\mathsf{CredProve}(params_{DC}, Nym_O, cred, sk_U, Nym_U, open_U, L)$. If $\mathsf{PKVerify}(params_{PK}, (Nym_O, \mathsf{Commit}(sk_U, 0)), cred)$ rejects, or if $Nym_U \neq \mathsf{Commit}(sk_U, open_U)$, abort. Return $credproof \leftarrow \mathsf{RandProof}((Nym_O, Nym_U), (0, open_U), cred)$.

$\mathsf{CredVerify}(params_{DC}, Nym_O, credproof, Nym_U, L)$ runs PKVerify.

$\mathsf{Issue}(params_{DC}, Nym_O, sk_I, Nym_I, open_I, cred, Nym_U, L)$
$\leftrightarrow$ $\mathsf{Obtain}(params_{DC}, Nym_O, sk_U, Nym_U, open_U, Nym_I, L)$. Abort if $L = 0$ and $Nym_O \neq Nym_I$. The issuer verifies *cred* using CredVerify and if it does not verify or if $Nym_I \neq \mathsf{Commit}(sk_I, open_I)$ or Nym_U is not a valid pseudonym, the issuer aborts. Else, the issuer and the user both compute $r_{L+1} = H(Nym_O, L+1)$. The issuer and the user run a two-party protocol with the following specifications: the public input is (Nym_I, Nym_U, r_{L+1}); the issuer's private input is $(sk_I, open_I)$ and the user's private input is $(sk_U, open_U)$. The output of the protocol is as follows: if $(sk_I, open_I)$ and $(sk_U, open_U)$ do not appropriately correspond to Nym_I, Nym_U, the protocol aborts; otherwise, the issuer receives no output while the user receives as output the value π computed as:

$$\pi \leftarrow \mathsf{NIZKPK}[sk_I \text{ in } Nym_I; sk_U \text{ in } \mathsf{Commit}(sk_U, 0)]\{(F(sk_I), F(sk_U), auth) : \\ \mathsf{VerifyAuth}(sk_I, (sk_U, r_{L+1}), auth)\} \ .$$

In Section 3.3 we give an efficient instantiation of such a 2PC protocol for the specific authentication and NIZKPK schemes we use.

If $L = 0$, then the user outputs $cred_U = \pi$. Otherwise, the issuer obtains $credproof_I \leftarrow \mathsf{CredProve}(params_{DC}, Nym_O, cred, sk_I, Nym_I, open_I, L)$ and sends it to the user. Let $S_U = \mathsf{Commit}(sk_U, 0)$. Intuitively, $credproof_I$ is a proof that the owner of Nym_I has a level L credential under public key Nym_O, while π is proof that the owner of Nym_I delegated to the owner of S_U. The user concatenates $credproof_I$ and π to obtain $credproof_I \circ \pi$. To get $cred_U$, U needs to project $credproof_I \circ \pi$ into a proof about (Nym_O, S_U) instead of Nym_I.

Remark 4. We can attach public attributes to each level of the credential. We compute $r_\ell = H(sk_O, \ell, \mathsf{attr}_1, \ldots, \mathsf{attr}_\ell)$, where attr_i is the *set* of attributes added by the ith delegator in the delegation chain. When the user shows or delegates a credential, he must display all the attributes associated with each level.

Message authentication scheme. Just like a signature scheme, an authentication scheme must be complete and unforgeable. For our application we need to strengthen the unforgeability property in two ways. First, we require F-*Unforgeability* [BCKL08], which guarantees that for some well-defined bijection F, no adversary can output $(F(\boldsymbol{m}), auth)$ without first getting an authenticator on m. (We write $F(\boldsymbol{m}) = F(m_1, \ldots, m_n)$ to denote $(F(m_1), \ldots, F(m_n))$.) Second we require a new property which we call *certification security*; the authenticator is unforgeable even if the adversary learns a signature on the challenge secret key. An authentication scheme is F-unforgeable and certification secure if for all ppt. adversaries $\mathcal{A}$ there exists negligible ν such that:

$$\begin{aligned}&\Pr[params_A \leftarrow \mathsf{AuthSetup}(1^k); sk \leftarrow \mathsf{AuthKg}(params_A);\\ &\quad(\boldsymbol{y}, auth) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Auth}}(params_A, sk, .), \mathcal{O}_{\mathsf{Certify}}(params_A, ., (sk, ., \ldots))}(params_A, F(sk)) :\\ &\quad\mathsf{VerifyAuth}(params_A, sk, F^{-1}(\boldsymbol{y}), auth) = 1 \wedge F^{-1}(\boldsymbol{y}) \notin Q_{\mathsf{Auth}}] \leq \nu(k) \ ,\end{aligned}$$

where the oracle $\mathcal{O}_{\mathsf{Auth}}(params_A, sk, \boldsymbol{m})$ outputs $\mathsf{Auth}(params_A, sk, \boldsymbol{m})$ and stores $\boldsymbol{m}$ on Q_{Auth}, and oracle $\mathcal{O}_{\mathsf{Certify}}(params_A, sk^*, (sk, m_2, \ldots, m_n))$ outputs the signature $\mathsf{Auth}(params_A, sk^*, (sk, m_2, \ldots, m_n))$.

Theorem 1. *Let* $\mathsf{AuthSetup}, \mathsf{AuthKg}, \mathsf{Auth}, \mathsf{VerifyAuth}$ *be an F-unforgeable certification-secure authentication scheme, H be a collision resistant hash function, and* $\mathsf{PKSetup}, \mathsf{PKProve}, \mathsf{PKVerify}$ *be a randomizable, partially extractable, composable zero-knowledge non-interactive proof of knowledge system. Then the above construction constitutes a secure anonymous delegatable credential scheme. See full version [BCC+ 08] for proof.*

We will construct our authentication scheme based the BB-CDH and BB-HSDH assumptions (defined in Section 3.3). Groth-Sahai proofs require either the SXDH assumption or the Decision Linear Assumption [GS08]. Our two party protocol requires a homomorphic encryption scheme.

3.3 Building Block Instantiations

Bilinear Maps and Assumptions. We use standard notation for groups with a computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$. See, e.g., [BLS04, GPS06]. The security of our scheme is based on strengthened versions of the SDH [BB04] and CDH assumptions. BB-CDH is implied by SDH; [Boy08] describes how to extend Generalized Diffie Hellman [BBG05] to cover these two assumptions and prove their generic group security.

Definition 3 (BB-HSDH). *Let $x, c_1 \ldots c_q \leftarrow Z_p$. On input $g, g^x, u \in G_1$, $h, h^x \in G_2$ and the tuple $\{g^{1/(x+c_\ell)}, c_\ell\}_{\ell=1\ldots q}$, it is computationally infeasible to output a new tuple $(g^{1/(x+c)}, h^c, u^c)$.*

Definition 4 (BB-CDH). *Let $x, y, c_1 \ldots c_q \leftarrow Z_p$. On input $g, g^x, g^y \in G_1$, $h, h^x \in G_2$ and the tuple $\{g^{1/(x+c_\ell)}, c_\ell\}_{\ell=1\ldots q}$, it is computationally infeasible to output g^{xy}.*

F-Unforgeable Certification Secure Message Authentication Scheme. Our authentication scheme is based on the Boneh-Boyen weak signature scheme [BB04], where $\mathsf{Sign}_{sk}(m) = g^{1/(sk+m)}$. Belenkiy et al. showed that the Boneh-Boyen signature scheme is F-unforgeable for the bijection $F(m) = (g^m, u^m)$ (under a very strong assumption), and that the Groth-Sahai proof system can be used to prove knowledge of such a signature. Boneh-Boyen signatures are not certification secure because $\mathsf{Sign}_{sk}(m) = \mathsf{Sign}_m(sk)$. We show how to achieve certification security; we also authenticate a vector of messages and weaken the underlying security assumption. The construction is as follows: $\mathsf{Auth}(sk, m_1||m_2)$ chooses random keys K^*, K_1, K_2 and returns $(\mathsf{Sign}_{sk}(K^*), \mathsf{Sign}_{K^*}(K_1), \mathsf{Sign}_{K^*}(K_2), \mathsf{Sign}_{K_1}(m_1), \mathsf{Sign}_{K_2}(m_2), F(K^*), F(K_1), F(K_2))$. At a high level, this construction eliminates any symmetries between $\mathsf{Auth}_{sk}(m)$ and $\mathsf{Auth}_m(sk)$. See full version [BCC+08] for details.

Theorem 2. *The message authentication scheme above is F-unforgeable and certification secure for $F(m_i) = (h^{m_i}, u^{m_i})$ under the BB-HSDH and BB-CDH assumptions.* See full version [BCC+08] for proof. The signature scheme obtained by setting $pk = h^{sk}$ may be of independent interest.

Commitment scheme. A commitment to $x \in Z_p$ consists of two GS commitments $\mathsf{GSCommit}(h^x, o_1)$, $\mathsf{GSCommit}(u^x, o_2))$ and a NIPK_{GS} proof that these are commitments to the same value x. This allows us to extract $F(x) = (h^x, u^x)$.

Proof of knowledge of an authenticator. We need a NIZKPK of an authenticator for messages $\boldsymbol{m} = (m_1, m_2)$, where the first value is hidden in commitment C_{m_1} and the second value m_2 is publicly known. In our notation, this is:

$$\mathsf{NIZKPK}[sk \text{ in } C_{sk}; m_1 \text{ in } C_{m_1}]\{(F(sk), F(m_1), auth) : \mathsf{VerifyAuth}(params_A, sk, (m_1, m_2), auth) = 1\}.$$

Since Boneh-Boyen signatures are verified using pairing product equations, we can use Groth-Sahai proofs, see full version [BCC+08] for details.

Creating a NIZKPK of an authenticator. The issuer chooses K^*, K_1, K_2 and can generate most of the proof. Then, the issuer and user need to jointly compute a NIZKPK of a Boneh-Boyen signature on the user's secret key. We outline the protocol, see full version [BCC+08] for details.

Let $\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}$ be an additively homomorphic semantically secure encryption scheme, let "$\oplus$" denote the homomorphic operation on ciphertexts; for e a ciphertext and r an integer, $e \otimes r$ denotes "adding" e to itself r times. The user with input m_1, and the issuer with input K_1 run the following protocol to compute $\mathsf{Sign}_{K_1}(m_1) = g^{1/(K_1+m_1)}$:

1. The issuer generates $(sk_{hom}, pk_{hom}) \leftarrow \mathsf{Keygen}(1^k)$ in such a way that the message space is of size at least $2^k p^2$. He then computes $e_1 = \mathsf{Enc}(pk_{hom}, K_1)$ and sends e_1, pk_{hom} to the user and engages with her in an interactive zero-knowledge proof that e_1 encrypts to a message in $[0, p]$.
2. The user chooses $r_1 \leftarrow Z_p$ and $r_2 \leftarrow \{0, \ldots, 2^k p\}$, then computes $e_2 = ((e_1 \oplus \mathsf{Enc}(pk_{hom}, m_1)) \otimes r_1) \oplus \mathsf{Enc}(pk_{hom}, r_2 p)$ and sends e_2 to the user.
3. The issuer and the user perform an interactive zero-knowledge proof in which the user shows that e_2 has been computed correctly using the message in C_{m_1}, and that r_1, r_2 are in the appropriate ranges.
4. The issuer decrypts $x = \mathsf{Dec}(sk_{hom}, e_2)$, sends the user $\sigma^* = g^{1/x}$.
5. The user computes $\sigma = {\sigma^*}^{r_1}$ and verifies that it is a correct weak BB signature on m_1. The issuer obtains no information about m_1.

Theorem 3. *The above is a secure two-party computation for computing Boneh-Boyen signatures.* (See full version [BCC+08] for proof.)

Acknowledgements. Jan Camenisch was supported in part by the European Commission through the ICT and IST programmes under contract ICT-216483 PRIMELIFE. Markulf Kohlweiss was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT and IST programmes under the following contracts: ICT-216483 PRIMELIFE and ICT-216676 ECRYPT II. Hovav Shacham was supported by an AFOSR MURI grant and, while at the Weizmann Institute of Science, by a Koshland Scholars Program postdoctoral fellowship. Mira Belenkiy, Melissa Chase and Anna Lysyanskaya acknowledge the support of NSF grants 0831293, 0627553 and 0347661. The information in this document reflects only the authors' views.

References

[Bar01] Barak, B.: Delegatable signatures. Technical report, Weizmann Institute of Science (2001)

[BB04] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

[BBG05] Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)

[BCC$^+$08] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. Cryptology ePrint Archive, Report 2008/428 (2008), http://eprint.iacr.org/2008/428

[BCKL08] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)

[BDI$^+$99] Burmester, M., Desmedt, Y., Itoh, T., Sakurai, K., Shizuya, H.: Divertible and subliminal-free zero-knowledge proofs for languages. Journal of Cryptology 12(3), 197–223 (1999)

[BDMP91] Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge. SIAM Journal of Computing 20(6), 1084–1118 (1991)

[BFM88] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC 1988 (1988)

[BLS04] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)

[Boy08] Boyen, X.: The uber-assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008)

[Bra99] Brands, S.: Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy. PhD thesis, Eindhoven Inst. of Tech. The Netherlands (1999)

[Cha85] Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)

[CHK$^+$06] Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: CCS 2006 (2006)

[CL01] Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Eurocrypt 2001 (2001)

[CL02a] Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 61. Springer, Heidelberg (2002)

[CL02b] Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)

[CL06] Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)

[CS97] Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)

[Dam02] Damgård, I.: On σ-protocols (2002), http://www.daimi.au.dk/~ivan/Sigma.ps

[DSY90] De Santis, A., Yung, M.: Cryptographic applications of the non-interactive metaproof and many-prover systems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 366–377. Springer, Heidelberg (1991)

[FLS99] Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SIAM Journal on Computing 29(1), 1–28 (1999)

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)

[Gol00] Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, New York (2000)

[GOS06] Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)

[GPS06] Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), `http://eprint.iacr.org/`

[GS02] Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[KP98] Kilian, J., Petrank, E.: An efficient non-interactive zero-knowledge proof system for np with general assumptions. J. of Cryptology (1998)

[LRSW99] Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, p. 184. Springer, Heidelberg (2000)

[SCP00] De Santis, A., Di Crescenzo, G., Persiano, G.: Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all NP relations. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, p. 451. Springer, Heidelberg (2000)

Computational Differential Privacy

Ilya Mironov[1], Omkant Pandey[2,⋆],
Omer Reingold[3,⋆,⋆⋆], and Salil Vadhan[4,⋆,⋆⋆]

[1] Microsoft Research, Silicon Valley Campus
mironov@microsoft.com
[2] University of California, Los Angeles
omkant@cs.ucla.edu
[3] Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel
omer.reingold@weizmann.ac.il
[4] School of Engineering and Applied Sciences and Center for Research on
Computation and Society, Harvard University
salil@eecs.harvard.edu

Abstract. The definition of differential privacy has recently emerged as a leading standard of privacy guarantees for algorithms on statistical databases. We offer several relaxations of the definition which require privacy guarantees to hold only against *efficient*—i.e., computationally-bounded—adversaries. We establish various relationships among these notions, and in doing so, we observe their close connection with the theory of pseudodense sets by Reingold et al. [1]. We extend the dense model theorem of Reingold et al. to demonstrate equivalence between two definitions (indistinguishability- and simulatability-based) of computational differential privacy.

Our computational analogues of differential privacy seem to allow for more accurate constructions than the standard information-theoretic analogues. In particular, in the context of private approximation of the distance between two vectors, we present a differentially-private protocol for computing the approximation, and contrast it with a substantially more accurate protocol that is only *computationally* differentially private.

1 Introduction

A curator of a statistical database may promote valuable social purposes in his operation. At the same time, non-careful procedures for managing access to the database may expose sensitive information (in potentially subtle ways), damaging individual contributors and putting the curator at the risk of legal liability.

The statistics, database, and datamining communities have long understood that there is a complicated space of possible trade-offs between usability of statistical databases and secrecy of individual records. A recent line of research in

⋆ Work done in part while visiting Microsoft Research Silicon Valley.
⋆⋆ Supported by US-Israel BSF grant 2006060.

privacy in statistical databases is focussed on formalizing and quantifying the notions of privacy and usability, and developing privacy-preserving analogues for many types of queries or algorithms one may want to run on a database (surveyed by Dwork [2]).

The cornerstone of the new approach to privacy is the definition of *differential privacy*, which first appeared in [3]. Intuitively, the definition captures the risk of joining the database, where the risk is measured as the adversary's success in predicting whether a single record is present in the database, given the rest of the database. The definition gives unconditional guarantees (including privacy for (small) groups) against a powerful adversary, preserved by sequential composition, and still allows many types of statistical or machine learning analyses, as shown in [4,5,6]. We note that the adversary's gain in success probability typically tolerated in applications of differential privacy is not zero or even "cryptographically" small (and cannot be so under any reasonable utility guarantees [3]).

The standard definition of differential privacy is very strong in that it provides privacy even against a computationally unbounded adversary. While there has been substantial success in designing mechanisms that achieve this strong definition (e.g., [5,6,7,8]), in this paper we suggest that such information-theoretic privacy may sometimes have a significant price (in utility or complexity). Thus we propose several computational analogues of differential privacy, where we only require privacy against a feasible (i.e., polynomial time) adversary.

Immediate benefits of the relaxation include combining pseudo-random generators with differentially-private mechanisms, or running such mechanisms in a distributed manner with only computational guarantees of security. More importantly, computational differentially-private mechanisms may exist for problems for which standard differentially-private mechanisms are impossible or unknown. We propose the problem of constructing a two-party protocol with two-sided guarantees of privacy for approximating the Hamming distance between two bit-vectors as a candidate for separating the power of computational and information-theoretic definitions of privacy.

Definition of computational differential privacy. There are two natural approaches to defining differential privacy with a computational flavor. The first one, which may be characterized as *indistinguishability*-based, goes back to the definition of differential privacy and replaces an unrestricted adversary with a computationally-bounded one. Doing so, at least in the non-uniform case, does not expand the class of privacy-preserving algorithms as the new definition can be shown equivalent to the old one. If instead, we start with the weaker definition of (ϵ, δ)-differential privacy [9], which allows some negligible additive distinguishing advantage, we do obtain a new class of mechanisms that are private under the new definition, which we call IND-CDP.

The second approach to defining computational differential privacy, which we naturally call *simulation*-based or SIM-CDP, builds upon the definition of differential privacy and its properties. It asserts that the view of the adversary can be simulated given access to a differentially-private function of the database (and thus the simulation is differentially private). The simulated view must be computationally indistinguishable from the real mechanism's transcript.

Given these two equally compelling definitional approaches it is quite natural to consider their relative power. We show that both definitions are closed under sequential composition and provide privacy for (small) groups of records. We thus switch to study the relationship between these definitions. One can easily demonstrate that SIM-CDP implies IND-CDP. The converse of this statement, however, is an intriguing question that we leave open in this work. Instead, in the main technical contribution of this paper we establish equivalence between a weaker (though still natural) simulation-based definition (called $\text{SIM}_{\forall\exists}$-CDP) and IND-CDP (Section 3). We also generalize our definition to interactive mechanisms, where we uncover one more definition, called SIM^{+}-CDP. A summary of our results relating various definitions of privacy is presented in Figure 1.

$$\begin{tikzcd}
\text{IND-CDP} \arrow[r, shift right, "\text{Theorem 3}"'] & \text{SIM}_{\forall\exists}\text{-CDP} \arrow[l, shift right, "\text{Theorem 1}"'] & \text{SIM-CDP} \arrow[l, "\text{immediate}"'] \arrow[r, shift right, "/" marking, "\text{Theorem 4}"'] & \text{SIM}^{+}\text{-CDP} \arrow[l, shift right, "\text{Theorem 4}"']
\end{tikzcd}$$

Fig. 1. Relations between definitions of computational differential privacy

Our approach to proving equivalence between IND-CDP and $\text{SIM}_{\forall\exists}$-CDP establishes a surprising connection between computational differential privacy and "pseudodense sets" studied by Reingold et al. [1], who were in turn motivated by the work of Green, Tao, and Ziegler in additive combinatorics [10,11] (closely related notions were previously studied by [12]). In fact, IND-CDP can be stated in terms of *pseudodensity* of the mechanism's distribution over adjacent datasets, and SIM-CDP is equivalent to existence of *models* that are *dense* for all adjacent pairs, to use the language of [1]. We extend the Dense Model Theorem of [1] to account for the symmetry of the definitions of differential privacy.

As mentioned above, we also construct protocols achieving computational differential privacy that seem to admit significantly better accuracy than any information-theoretically private protocols. Our main example is a private approximation of the Hamming distance between two vectors in a two-party setting. We propose three protocols for this problem: one protocol with information-theoretic differential privacy guarantees and multiplicative approximation error, and two protocols handling the semi-honest and malicious cases achieving computational differential privacy (specifically SIM-CDP) and with error *independent of the size of the input* (it only depends on the privacy and security parameters). Subsequent to our results, this gap in accuracy between information-theoretic and computational differential privacy was shown to be inherent [13].

Secure vs differentially-private computations. The problem of secure Hamming distance computation, together with closely related problems of secure scalar product and secure set-intersection cardinality [14,15,16,17,18], may benefit from casting them in the differential privacy framework. Indeed, the standard cryptographic guarantee of letting the parties compute the output of a function, such as the Hamming distance between two vectors, while hiding everything else about their inputs may be insufficient to argue security of a sequential composition of this protocol when one has to consider the information leaked through the *output* of the function. For example, if Alice varies her input while Bob's vector stays constant, by observing the output of the protocol Alice may learn individual values of Bob's input bits. Differential privacy treatment addresses the orthogonal question of *what* is computed rather than *how*; in particular, it may be used to analyze effects of adaptive sequential or concurrent composition on the adversary's confidence in predicting any particular bit, even in the presence of auxiliary information.

2 Definitions

We describe our definitions in this section. We start by introducing some notation.

Mechanism f. In our definitions, we will be interested in measuring privacy guarantees provided by randomized mechanisms, denoted f. Mechanism f operates on subsets D of a (potentially infinite) universe U, which we associate with databases, and outputs a value in the range $\mathcal{R}$. The size of the input D will be denoted by n. We say that two databases D and D' are *adjacent* if their symmetric difference contains at most one record (i.e., $|D \Delta D'| \leq 1$)[1]. Further, the maximum size of the output of f is m.

As we are dealing with computational notion, we will mostly be concerned with *efficient* adversaries. Unless specified otherwise, throughout the paper, an *efficient* adversary is modeled by a family of polynomial-sized circuits $\{A_\kappa\}_{\kappa \in \mathbb{N}}$, or equivalently, a nonuniform probabilistic polynomial time (PPT) Turing machine.

Parameter κ. A "security" parameter κ controls various quantities in our definitions/constructions as follows. The size of the adversary will be polynomial in κ. The mechanism is parameterized by κ, which lets us consider a family $\{f_\kappa\}_{\kappa \in \mathbb{N}}$, where $f_\kappa \colon \mathcal{D} \to \mathcal{R}_\kappa$. The output size m of f is required to be (at most) polynomial in κ. We say that a function in κ is *negligible* if it approaches zero faster than the reciprocal of any polynomial in κ.

We first recall the standard definition of ϵ-differential privacy [3]:

Definition 1 (ϵ-DP privacy). *A randomized mechanism $f \colon \mathcal{D} \to \mathcal{R}$ provides ϵ-DP if for all adjacent inputs $D, D' \in \mathcal{D}$ (i.e., $|D \Delta D'| \leq 1$) and all subsets $S \subseteq \mathcal{R}$*

$$\Pr[f(D) \in S] \leq e^{\epsilon} \times \Pr[f(D') \in S],$$

where the probability space is f's coin tosses.

[1] Δ denotes symmetric difference of two sets and $|\cdot|$ denotes the size when the argument is a set.

A closely related notion of (ϵ, δ)-differential privacy [9] has an additive parameter that allows the probabilities to diverge when they are both relatively small:

Definition 2 **((ϵ, δ)-DP privacy).** *A randomized mechanism $f \colon \mathcal{D} \to \mathcal{R}$ provides (ϵ, δ)-DP if for all adjacent inputs D and D' and all subsets $S \subseteq \mathcal{R}$*

$$\Pr[f(D) \in S] \leq e^{\epsilon} \times \Pr[f(D') \in S] + \delta,$$

where the probability space is f 's coin tosses.

Our first new definition, ϵ_κ-IND-CDP, is an adaptation of ϵ-differential-privacy to the computational setting. This adaptation is obtained by considering a PPT adversary A, and requiring that f "looks differentially private" to every such A.

Definition 3 **(IND-CDP privacy).** *An ensemble $\{f_\kappa\}_{\kappa \in \mathbb{N}}$ of* randomized *functions $f_\kappa \colon \mathcal{D} \to \mathcal{R}_\kappa$ provides ϵ_κ-IND-CDP if there exists a negligible function* negl$(\cdot)$ *such that for every nonuniform* PPT TM *("distinguisher") A, every polynomial $p(\cdot)$, every sufficiently large $\kappa \in \mathbb{N}$, all data sets $D, D' \in \mathcal{D}$ of size at most $p(\kappa)$ such that $|D \varDelta D'| \leq 1$, and every advice string z_κ of size at most $p(\kappa)$, it holds that*

$$\Pr\left[A_\kappa(f_\kappa(D)) = 1\right] \leq e^{\epsilon_\kappa} \times \Pr\left[A_\kappa(f_\kappa(D')) = 1\right] + \mathrm{negl}(\kappa),$$

where we write $A_\kappa(x)$ for $A(1^\kappa, z_\kappa, x)$ and the probability is taken over the randomness of mechanism f_κ and adversary A_κ.

Notice that if the adversary A is allowed unbounded computation time, then the definition simply says that for any fixed κ the mechanism f_κ is $(e_\kappa, \delta_\kappa)$-DP for δ_κ being negl(κ). The reason we do not consider the computational analogue of ϵ_κ-DP (with $\delta = 0$) is that it ends up being equivalent to information-theoretic ϵ-DP. Indeed, for any singleton $r \in \mathcal{R}_\kappa$, we can choose A_κ to be the indicator function for $\{r\}$, implying that $\Pr[f_\kappa(D) = r] \leq e^{\epsilon_\kappa} \Pr[f_\kappa(D') = r]$. This immediately implies ϵ_κ-DP by summing both sides over all $r \in S$ for any subset $S \subset \mathcal{R}_\kappa$.

Our second definition, ϵ_κ-SIM-CDP, is described next. This definition interprets "looks differentially private" differently from our first definition: it says that f "looks differentially private" if there exists an ϵ-DP mechanism F (called simulator) such that $F(D)$ and $f(D)$ are computationally indistinguishable for every D.

Definition 4 **(SIM-CDP privacy).** *An ensemble $\{f_\kappa\}_{\kappa \in \mathbb{N}}$ of randomized functions $f_\kappa \colon \mathcal{D} \to \mathcal{R}_\kappa$ provides ϵ_κ-SIM-CDP if there exists an ensemble $\{F_\kappa\}_{\kappa \in \mathbb{N}}$ of ϵ_κ*-differentially-private mechanisms *$F_\kappa \colon \mathcal{D} \to \mathcal{R}_\kappa$ and a negligible function* negl$(\cdot)$*, such that for every non-uniform* PPT TM *A, every polynomial $p(\cdot)$, every sufficiently large $\kappa \in \mathbb{N}$, every data set $D \in \mathcal{D}$ of size at most $p(\kappa)$, and every advice string z_κ of size at most $p(\kappa)$, it holds that,*

$$\left|\Pr\left[A_\kappa(f_\kappa(D)) = 1\right] - \Pr\left[A_\kappa(F_\kappa(D)) = 1\right]\right| \leq \mathrm{negl}(\kappa).$$

That is, $f_\kappa(D)$ and $F_\kappa(D)$ are computationally indistinguishable.

Note that the definition does not require F to be computable in probabilistic polynomial time; it only has to exist, and be (information theoretically) differentially private.

The definition of SIM-CDP requires that there *exists* a simulator F that acts in a differentially-private manner on *all* pairs of adjacent inputs. This suggests a weakening of the definition, where the order of quantifiers is reversed, i.e., instead of requiring a global simulator that works for all pairs of databases, we require that for *any* pair of adjacent databases there *exists* a simulator whose distributions on these two inputs satisfy the differential privacy condition.

Definition 5 (SIM$_{\forall\exists}$-CDP privacy). *An ensemble $\{f_\kappa\}_{\kappa\in\mathbb{N}}$ of randomized functions $f_\kappa\colon \mathcal{D} \to \mathcal{R}_\kappa$ provides ϵ_κ-SIM$_{\forall\exists}$-CDP if for all polynomials $p(\cdot)$, all sequences $\{(D_\kappa, D'_\kappa)\}_{\kappa\in\mathbb{N}}$ of pairs of datasets such that $|D_\kappa| \le p(\kappa), |D'_\kappa| \le p(\kappa)$ and $|D_\kappa \Delta D'_\kappa| \le 1$, there exist ensembles $\{F_\kappa(D_\kappa)\}_{\kappa\in\mathbb{N}}$ and $\{F_\kappa(D'_\kappa)\}_{\kappa\in\mathbb{N}}$, such that the following two conditions hold:*

1. *[F_κ is ϵ_κ-DP.] For all subsets $S \subset \mathcal{R}$:*

$$e^{-\epsilon_\kappa} \times \Pr[F_\kappa(D'_\kappa) \in S] \le \Pr[F_\kappa(D_\kappa) \in S] \le e^{\epsilon_\kappa} \times \Pr[F_\kappa(D'_\kappa) \in S].$$

2. *[$f_\kappa(D_\kappa), f_\kappa(D'_\kappa)$ are indistinguishable from $F_\kappa(D_\kappa), F_\kappa(D'_\kappa)$ respectively.] For every non-uniform PPT TM A, every polynomial $q(\cdot)$, every sufficiently large $\kappa \in \mathbb{N}$, and every advice string z_κ of size at most $q(\kappa)$:*

$$|\Pr[A_\kappa(f_\kappa(D)) = 1] - \Pr[A_\kappa(F_\kappa(D)) = 1]| \le \mathrm{negl}(\kappa) \text{ for } D \in \{D_\kappa, D'_\kappa\}$$

where we write $A_\kappa(x)$ for $A(1^\kappa, z_\kappa, x)$.

We may also consider an even weaker definition, where the probability $\Pr[F_\kappa(D_\kappa) \in S]$ is only bounded from above by $e^\epsilon \times \Pr[F_\kappa(D'_\kappa) \in S]$ (and a second pair of simulators exist for the ordered (D'_κ, D_κ) pair), but as we shall see in Section 3, it turns out to be equivalent to SIM$_{\forall\exists}$-CDP.

Robustness of our definitions. Protocols satisfying our definitions retain their privacy guarantees under sequential composition, and for all but SIM$_{\forall\exists}$-CDP we directly prove group privacy (for records); in both cases the privacy parameters ϵ and δ deteriorate linearly with the number of compositions or records, respectively. Due to space constraints, a detailed exposition is presented in the full version.Informally, under the definition of group privacy the adversary has to guess whether two or more elements are simultaneously present or absent in the database, given the rest of the database. The definition of group privacy is often applicable when the differentially-private mechanism is preceded by computations that may amplify (up to a constant) the number of records affected by any individual [19].

Interactive case. For simplicity, current definitions consider only non-interactive mechanisms. An extension to the interactive case will be presented in Section 4. As it turns out, a variation of ϵ_κ-SIM-CDP, called ϵ_κ-SIM$^+$-CDP, can also be defined and proven separate from ϵ_κ-SIM-CDP (see Section 4).

3 Relations Among Various Notions of CDP

In this section we establish reductions between the three definitions of computational differential privacy. The first implication, namely, that SIM-CDP implies SIM$_{\forall\exists}$-CDP, which implies IND-CDP, is the easiest (Section 3.1). The proof that IND-CDP implies SIM$_{\forall\exists}$-CDP (and thus that the two definitions are equivalent) extends the Dense Model Theorem of [1] and is significantly more involved (Section 3.2).

3.1 Simulatability Implies Indistinguishability

Theorem 1 (SIM-CDP $\Rightarrow$ SIM$_{\forall\exists}$-CDP $\Rightarrow$ IND-CDP). *If an ensemble $\{f_\kappa\}_{\kappa\in\mathbb{N}}$ of randomized functions $f_\kappa\colon \mathcal{D}\to\mathcal{R}_\kappa$ provides ϵ_κ-SIM-CDP, then it also provides ϵ_κ-SIM$_{\forall\exists}$-CDP; if it provides ϵ_κ-SIM$_{\forall\exists}$-CDP, it also provides ϵ_κ-IND-CDP.*

Proof. The first implication is by construction, the second follows by a hybrid argument. The full proof appears in the full version. □

In the section that follows, we will prove that ϵ_κ-IND-CDP $\Rightarrow$ ϵ_κ-SIM$_{\forall\exists}$-CDP by giving an extension of the Dense Model Theorem of [1] (which may be of independent interest).

3.2 Dense Sets and IND-CDP $\Rightarrow$ SIM$_{\forall\exists}$-CDP

First, we define or recall notions of (non-uniform) density, pseudodensity, and indistinguishability for distributions, closely following [1].

Consider two distributions X and Y defined over $\mathcal{R}$, and a collection $\mathcal{A}$ of randomized predicates $A\colon\mathcal{R}\to\{0,1\}$, which may be, for instance, all circuits of size at most $s(\kappa)$, where κ is the security parameter.

We say that X is e^ϵ-*dense* in Y if

$$\forall x\in\mathcal{R}\qquad \Pr[X=x]\le e^\epsilon\cdot\Pr[Y=x].$$

We define X as δ-*indistinguishable* from Y with respect to $\mathcal{A}$ if

$$\forall A\in\mathcal{A}\qquad |\Pr[A(X)=1]-\Pr[A(Y)=1]|\le\delta,$$

where here and elsewhere in this section we write $A(X)$ for the distribution on the range of A obtained by applying A to the variable sampled according to X and the probability space is that of X and A's coins.

Finally, a "combination" of the two definitions is rather naturally defined as X being (e^ϵ,δ)-*pseudodense* in Y with respect to $\mathcal{A}$ if

$$\forall A\in\mathcal{A}\qquad \Pr[A(X)=1]\le e^\epsilon\cdot\Pr[A(Y)=1]+\delta.$$

The connections between notions of differential privacy, SIM-CDP and IND-CDP and the above definitions are immediate:

A randomized mechanism $f\colon \mathcal{D} \to \mathcal{R}$ is ϵ_κ-DP if and only if $f(D)$ is e^{ϵ_κ}-dense in $f(D')$ for all adjacent pairs D and D', where the probability space of the distributions $f(D)$ and $f(D')$ over $\mathcal{R}$ is f's randomness.

An ensemble $\{f_\kappa\}_{\kappa\in\mathbb{N}}$ is ϵ_κ-IND-CDP if and only if there is a super-polynomial function $s(\kappa) = \kappa^{\omega(1)}$ such that for all sufficiently large κ, all adjacent pairs $D, D' \in \mathcal{D}$ of size at most $s(\kappa)$, the distribution $f_\kappa(D)$ is $(e^{\epsilon_\kappa}, \frac{1}{s(\kappa)}))$-pseudodense in $f_\kappa(D')$, with respect to the set $\mathcal{A}_\kappa$ of circuits of size at most $s(\kappa)$.

Similarly, $\{f_\kappa\}_{\kappa\in\mathbb{N}}$ is ϵ_κ-SIM-CDP if there exists an ensemble $\{F_\kappa\}_{\kappa\in\mathbb{N}}$ and a super-polynomial function $s(\kappa) = \kappa^{\omega(1)}$ such that all randomized mechanisms $F_\kappa\colon \mathcal{D} \to \mathcal{R}_\kappa$ are ϵ_κ-DP and for all sufficiently large κ, all $D \in \mathcal{D}$ of size at most $s(\kappa)$, distributions $f_\kappa(D)$ and $F_\kappa(D)$ are $\frac{1}{s(\kappa)}$-indistinguishable for the set $\mathcal{A}_\kappa$ of circuits of size at most $s(\kappa)$.

It is convenient to consider the two-sided notions of *mutually e^ϵ-dense* and *mutually (e^ϵ, δ)-pseudodense* sets, where X and Y are e^ϵ-dense (resp., (e^ϵ, δ)-pseudodense) in each other. Since the definitions of IND-CDP and SIM-CDP are symmetric in terms of the databases D and D', all relationships between distributions of f, f_κ, and F_κ on D and D' in the formulations above are, in fact, mutual.

Reingold et al. [1] showed that pseudodensity is indeed a composition of density and indistinguishability for some classes of distinguishers. One implication is immediate: If there are X, Y, and M over $\mathcal{R}$ such that M is e^ϵ-dense in Y and X is δ-indistinguishable from M, then X is (e^ϵ, δ)-pseudodense in Y (all–with respect to the same class $\mathcal{A}$ of distinguishers). The first claim of the following theorem establishes the converse (with a caveat that indistinguishability is required to hold with respect to a class of functions of slightly higher complexity, as is common in proofs by reduction). The second claim is new to our work, and is the key to relating IND-CDP and SIM$_{\forall\exists}$-CDP.

Theorem 2. *Let X and Y be distributions over a finite universe $\mathcal{R}$ such that X is (e^ϵ, δ)-pseudodense in Y with respect to the family $\mathcal{T}(\mathcal{A})$ defined below.*

Claim I. *There exists a distribution M over $\mathcal{R}$ such that M is e^ϵ-dense in Y and X is 4δ-indistinguishable from M with respect to the family $\mathcal{A}$.*

Claim II. *Furthermore, if Y is e^ϵ-dense in X, then it can also be guaranteed that Y is e^ϵ-dense in M (i.e., Y and M are mutually e^ϵ-dense).*

If $\mathcal{A}$ is a family of predicates, we define $\mathcal{T}(\mathcal{A})$ as the collection of functions of the following type:

$$b(x) = \begin{cases} 1 & \text{if } h_1(x) + \cdots + h_k(x) > t; \\ 0 & \text{otherwise,} \end{cases}$$

where $h_i \in \mathcal{A} \cup \bar{\mathcal{A}}$, $t \in \mathbb{N}$, and $k = O(1/\delta^2 \log(e^\epsilon/4\delta)))$. $\bar{\mathcal{A}}$ is the set of negations of $\mathcal{A}$.

Proof. **Claim I.** The proof of Claim I appears in [1], where it is stated for the case when Y is the uniform distribution (but the proof generalizes to arbitrary Y).

Claim II. Assume towards a contradiction that for any M that is mutually e^ϵ-dense in Y there is a function A_M from $\mathcal{A}$ that distinguishes it from X with probability more than $\mu = 4\delta$. Note that the same automatically holds for M that is a convex combination of distributions that are mutually e^ϵ-dense in Y, because the set of such distributions is convex. By the min-max principle of game theory, or equivalently, duality of linear programming, there exists a convex combination $\bar{b}$ of functions from $\mathcal{A} \cup \bar{\mathcal{A}}$ that distinguishes *any* such M from X:

$$\Pr[\bar{b}(X) = 1] > \Pr[\bar{b}(M) = 1] + \mu. \tag{1}$$

The function $\bar{b}$ can be viewed as a distribution over predicates in $\mathcal{A} \cup \bar{\mathcal{A}}$.

Arrange elements x of $\mathcal{R}$ in the order of decreasing $\Pr[\bar{b}(x) = 1]$. Choose the set $S \subset \mathcal{R}$ as the initial part of the list so that $\Pr[Y \in S] = 1/(1 + e^\epsilon)$.[2]

Define Y_S as follows:

$$\Pr[Y_S = y] = \Pr[Y = y] \cdot \begin{cases} e^\epsilon & \text{if } y \in S; \\ e^{-\epsilon} & \text{otherwise.} \end{cases}$$

It is easy to verify that Y_S is a distribution:

$$\sum_{y \in \mathcal{R}} \Pr[Y_S = y] = e^\epsilon \sum_{y \in S} \Pr[Y = y] + e^{-\epsilon} \sum_{y \notin S} \Pr[Y = s]$$
$$= e^\epsilon \frac{1}{1 + e^\epsilon} + e^{-\epsilon} \frac{e^\epsilon}{1 + e^\epsilon} = 1.$$

By construction Y_S and Y are e^ϵ-dense in each other, and therefore Y_S can be distinguished from X by $\bar{b}$ with probability at least μ (think of Y_S as the "hardest" distribution for $\bar{b}$ from among those that are mutually e^ϵ-dense in Y).

We make use of the following lemma proved in [20]:

Lemma 1 ([20, Claim 2.3]). *Let $F\colon X \to [0, 1]$ be a bounded function, let Z and W be distributions such that $\mathbb{E}[F(Z)] \geq \mathbb{E}[F(W)] + \mu$. Then there is a real number $t \in [\mu/2, 1]$ such that*

$$\Pr[F(Z) \geq t] \geq \Pr[F(W) \geq t - \mu/2] + \mu/2.$$

Applying the lemma to $F(x) = \Pr[\bar{b}(x) = 1]$, X and Y_S, there exists a real t so that a deterministic function b defined as

$$b(x) = \begin{cases} 1 & \text{if } \Pr[\bar{b}(x) = 1] \geq t + \mu/2; \\ 0 & \text{if } \Pr[\bar{b}(x) = 1] \leq t; \\ \perp & \text{otherwise,} \end{cases}$$

[2] If exact equality cannot be achieved here, we take S to be the largest initial of the list such that $\Pr[Y \in S] < 1/(1 + e^\epsilon)$, and for the next element r of the list, define $\Pr[Y_S = r] \in \big[\Pr[Y = r]e^{-\epsilon}, \Pr[Y = r]e^\epsilon\big]$ in order to make Y_S a distribution.

is such that $\Pr[b(X) = 1] > \Pr[b(Y_S) \neq 0] + \mu/2$. In other words, classifying $x \in \mathcal{R}$ as "X" when $b(x) = 1$ and "Y_S" when $b(x) = 0$ is a good distinguisher between X and Y_S (Notice that there is some slack left between $b(x) = 1$ and $b(x) = 0$).

We claim that $b(y) = 0$ for all $y \in S$. Assume the opposite. By construction of the set S, $b(y) \neq 0$ for all $y \in S$. Since Y is e^{ϵ}-dense in X (this is the only time we use this condition), for all $y \notin S$ it holds that $\Pr[Y_S = y] = e^{-\epsilon}\Pr[Y = y] \leq e^{-\epsilon}e^{\epsilon}\Pr[X = y]$, i.e., the density of X dominates the density of Y_S outside S, including the set where b is zero. Therefore

$$\Pr[b(Y_S) = 0] = \sum_{y \notin S, b(y)=0} \Pr[Y_S = y] \leq \sum_{y \notin S, b(y)=0} \Pr[X = y] = \Pr[b(X) = 0],$$

which contradicts the fact that $\Pr[b(X) = 1] > \Pr[b(Y_S) \neq 0] + \mu/2$.

Now we know that $b(y) = 0$ outside S and we conclude that

$$\Pr[b(Y) \neq 0] = \Pr[b(Y_S) \neq 0] \cdot e^{-\epsilon} < (\Pr[b(X) = 1] - \mu/2) \cdot e^{-\epsilon}.$$

That is,

$$\Pr[b(X) = 1] > e^{\epsilon} \cdot \Pr[b(Y) \neq 0] + \mu/2. \tag{2}$$

This would contradict the pseudodensity condition except that b is not part of the family of functions $\mathcal{T}(\mathcal{A})$. The following lemma approximates b with a function from $\mathcal{T}(\mathcal{A})$:

Lemma 2 ([20, Claim 2.4]). *Let $F\colon \Omega \to [0;1]$ be a convex combination of bounded functions from a class G, let Z_1, Z_2 be two distributions on Ω, and let $\alpha, \beta > 0$. Then there are functions $f_1, \ldots, f_k \in G$ (not necessarily distinct) where $k = O(1/\alpha^2 \cdot \log(1/\beta))$, such that*

$$\Pr\left[\left|F(Z_i) - \frac{1}{k}(f_1(Z_i) + \cdots + f_k(Z_i))\right| > \alpha\right] \leq \beta \textit{ for } i = 1, 2.$$

We apply the lemma with parameters $\alpha = \mu/10$, $\beta = e^{-\epsilon}\mu/10$, and $F = \bar{b}$, we find an approximation to b with a function $\tilde{b}$ from $\mathcal{T}(\mathcal{A})$ with the property that

$$\Pr[\tilde{b}(X) = 1] \geq \Pr[b(X) = 1] - e^{-\epsilon}\mu/10,$$
$$\Pr[\tilde{b}(Y) = 1] \leq \Pr[b(Y) \neq 0] + e^{-\epsilon}\mu/10,$$

Combining these with equation (2) contradicts the pseudodensity of X in Y. Since we only consider predicates (0-1 functions), the threshold value t can be taken as an integer. □

Observe that if $\mathcal{A}_\kappa$ is the set of circuits of size $s(\kappa)$ for some $s(\kappa) = \kappa^{\omega(1)}$ and we take $\delta = 1/s(\kappa), \epsilon_\kappa \leq s(\kappa)$, then $\mathcal{T}(\mathcal{A})$ consists of circuits of size at most $t(\kappa) = s(\kappa)^{O(1)}$.

By applying both claims of Theorem 2, we obtain equivalence between the notions of IND-CDP and SIM$_{\forall\exists}$-CDP.

Theorem 3. *If a family of randomized mechanisms* $\{f_\kappa\}\colon \mathcal{D} \to \mathcal{R}_\kappa$ *is* ϵ_κ-IND-CDP *for* $\epsilon_\kappa \in O(\log\kappa)$*, it is also* ϵ_κ-SIM$_{\forall\exists}$-CDP.

Proof. If $\{f_\kappa\}$ is ϵ_κ-IND-CDP, then there is a super-polynomial function $s(\kappa) = \kappa^{\omega(1)}$ such that for all sufficiently large κ, and $D, D' \in \mathcal{D}$ of size at most $s(\kappa)$ and $|D_\kappa \varDelta D'_\kappa| \le 1$ the distribution $f_\kappa(D_\kappa)$ is $(e^{\epsilon_\kappa}, \frac{1}{s(\kappa)}))$-pseudodense in $f_\kappa(D')$, with respect to the set $\mathcal{A}_\kappa$ of circuits of size at most $s(\kappa)$. Let D, D' be adjacent data sets of size at most $s(\kappa)$. The pairs of distributions $f_\kappa(D)$ and $f_\kappa(D')$, where $f_\kappa(D)$ is $(e^\epsilon, 1/t(\kappa))$-pseudodense in $f_\kappa(D')$, are in situation of Claim I of Theorem 2. Therefore there exists a family of distributions $\{F_\kappa(D)\}_{\kappa\in\mathbb{N}}$ such that (a) $F_\kappa(D)$ and $f_\kappa(D)$ are $1/t(\kappa)^{\varOmega(1)}$-indistinguishable for circuits of size $t(\kappa)^{\varOmega(1)}$, and (b) $F_\kappa(D)$ is e^{ϵ_κ}-dense in $f_\kappa(D')$.

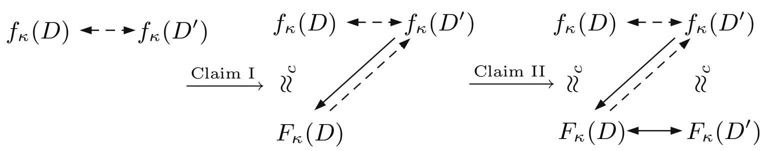
Since, in turn, $f_\kappa(D')$ is $(e^{\epsilon_\kappa}, 1/t(\kappa))$-pseudodense in $f_\kappa(D)$, which is indistinguishable from $F_\kappa(D)$, then $f_\kappa(D')$ is $(e^{\epsilon_\kappa}, 1/t(\kappa)^{\varOmega(1)})$-pseudodense in $F_\kappa(D)$ Indeed, for circuits $\{A_\kappa\}$ of size $t(\kappa)^{\varOmega(1)}$, we have

$$\begin{aligned}\Pr[A_\kappa(f_\kappa(D')) = 1] &\le e^{\epsilon_\kappa} \cdot \Pr[A_\kappa(f_\kappa(D)) = 1] + 1/t(\kappa) \le \\ &e^{\epsilon_\kappa} \cdot \Big(\Pr[A_\kappa(F_\kappa(D)) = 1] + 1/t(\kappa)^{\varOmega(1)}\Big) + 1/t(\kappa) \\ &\qquad = e^{\epsilon_\kappa} \cdot \Pr[A_\kappa(F_\kappa(D)) = 1] + 1/t(\kappa)^{\varOmega(1)},\end{aligned}$$

where the last part uses the conditions $\epsilon_\kappa = O(\log\kappa)$ and $t(\kappa) = \kappa^{\omega(1)}$.

Thus, we are in the situation of Claim II of Theorem 2 (two distributions, which are dense and pseudodense in one another). Therefore there exists a family of distributions $\{F_\kappa(D')\}_{\kappa\in\mathbb{N}}$, such that they are mutually e^{ϵ_κ}-dense in $\{F_\kappa(D)\}_{\kappa\in\mathbb{N}}$ and are $1/s(\kappa)$-indistinguishable from $\{f_\kappa(D')\}_{\kappa\in\mathbb{N}}$ by circuits of size $s(\kappa)$ for $s(\kappa) = \big(t(\kappa)^{\varOmega(1)}\big)^{\varOmega(1)} = t(\kappa)^{\varOmega(1)}$.

Pictorially the proof of the theorem is represented in Figure 2. □



$$\begin{tikzcd}
f_\kappa(D) \arrow[r, dashed, leftrightarrow] & f_\kappa(D') & & f_\kappa(D) \arrow[r, dashed, leftrightarrow] & f_\kappa(D') \arrow[dl, shift left, dashed] & & f_\kappa(D) \arrow[r, dashed, leftrightarrow] & f_\kappa(D') \arrow[dl, shift left, dashed] \\
& & \xrightarrow{\text{Claim I}} & \wr & & \xrightarrow{\text{Claim II}} & \wr & \wr \\
& & & F_\kappa(D) \arrow[ur, shift left] & & & F_\kappa(D) \arrow[ur, shift left] \arrow[r, leftrightarrow] & F_\kappa(D')
\end{tikzcd}$$

Fig. 2. Schematic proof of Theorem 3. $X \dashleftarrow Y$ means X is pseudodense in Y, $X \leftarrow Y$ means X is dense in Y. Claim I of Theorem 2 is applied to the pair $f_\kappa(D)$ and $f_\kappa(D')$; Claim II is applied to the pair $F_\kappa(D)$ and $f_\kappa(D')$.

4 Privacy-Preserving Two-Party Computation

We now extend our definitions to the interactive case. We work in the general two-party computation setting. A motivating scenario for a two-party "private" computation involves two hospitals H_1, H_2 (holding patient records D_1, D_2) who would like to compute some statistical function $h(D_1, D_2)$. Both hospitals are concerned about the privacy of patient records, and may not be willing or even legally allowed to share data.

Differentially-private multi-party computation (MPC) was considered by Beimel et al. [21], who mainly studied the efficiency trade-offs of the following natural paradigm for differentially-private computation of a function h: design an ϵ-DP mechanism $\widehat{h}$ that approximates h and then do secure MPC computation to obtain $\widehat{h}(D_1, D_2)$. [21] work only in the semi-honest/honest-majority models as it allows them to use information-theoretic MPC, which fits well with differential privacy.

The case of two-party computation (TWO-PC), however, is somewhat trickier as information-theoretically secure computation is impossible for generic functionalities [22]. Hence, one must resort to the computational security which interferes with the (standard) information-theoretic notion of differential privacy.

Dwork et al. [9] present a multi-party protocol run on top of a verifiable secret sharing scheme. Depending on the availability of secret channels, the protocol may only be secure against a computationally bounded adversary; however, no definition of computational differential privacy is given.

4.1 Definitions

We will now present our definitions for interactive protocols defined using *interactive functions* [23]. The reason for this choice (instead of interactive Turing machines) is that the concept of differential privacy is *orthogonal* to the choice of the computational model. In addition, many useful privacy mechanisms *may not* necessarily be efficiently computable (e.g., noise calibrated to smooth sensitivity [24] or exponential mechanisms [7]). Of course, when considering our computational definitions, we will require that the function corresponding to the *adversary* be implementable using a non-uniform PPT interactive Turing machine.

Notation. For ensembles $\{f_\kappa\}_{\kappa\in\mathbb{N}}$ and $\{g_\kappa\}_{\kappa\in\mathbb{N}}$ of randomized interactive functions f_κ, g_κ respectively, $\{\langle f_\kappa, g_\kappa\rangle\}_{\kappa\in\mathbb{N}}$ will denote the ensemble of interactive protocols defined by them. Further, in an execution $\langle f_\kappa, g^*_\kappa\rangle$ with inputs $x \in \mathcal{D}$ for the honest party, we will denote the view of the adversary (defined by interactive function g^*_κ) by $\mathtt{VIEW}_{\kappa,g^*_\kappa}(x)$.

Informally, a function ensemble $\{g_\kappa\}_{\kappa\in\mathbb{N}}$ is said to be an ensemble of *efficiently computable randomized interactive functions* if every function g_κ in the ensemble can be computed by a (non-uniform) PPT TM (a formal definition can be found in the full version). We now present our definitions. For an efficiently computable randomized interactive function g_κ, let $[g_\kappa]$ denote the binary string representing the interactive (non-uniform) Turing machine (equivalently, circuit) that implements g_κ.

Definition 6. *An ensemble $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ of interactive protocols,* ensures for $\{f_\kappa\}_{\kappa\in\mathbb{N}}$,

- ϵ_κ-DP*, if for every ensemble $\{g^*_\kappa\}_{\kappa\in\mathbb{N}}$ of randomized interactive functions, it holds that the ensemble $\{\mathtt{VIEW}_{\kappa,g^*_\kappa}(x)\}_{\kappa\in\mathbb{N}}$ provides* ϵ_κ-DP *with respect to $x \in \mathcal{D}$.*
- ϵ_κ-IND-CDP*, if for every ensemble $\{g^*_\kappa\}_{\kappa\in\mathbb{N}}$ of* efficiently computable *randomized interactive functions, and all sufficiently large κ, it holds that the ensemble $\{\mathtt{VIEW}_{\kappa,g^*_\kappa}(x)\}_{\kappa\in\mathbb{N}}$ provides* ϵ_κ-IND-CDP *(as per definition 3) with respect to $x \in \mathcal{D}$.*
- ϵ_κ-SIM-CDP*, if for every ensemble $\{g^*_\kappa\}_{\kappa\in\mathbb{N}}$ of* efficiently computable *randomized interactive functions, there exists an ensemble $\{F_\kappa\}_{\kappa\in\mathbb{N}}$ of ϵ_κ-*differentially-private mechanisms *$F_\kappa(\cdot)$ such that for every $x \in \mathcal{D}$, the probability ensembles $\{\mathtt{VIEW}_{\kappa,g^*_\kappa}(x)\}_{\kappa\in\mathbb{N}}$ and $\{F_\kappa(x)\}_{\kappa\in\mathbb{N}}$ are computationally indistinguishable.*

All three notions are defined symmetrically for the other ensemble $\{g_\kappa\}_{\kappa\in\mathbb{N}}$.

A protocol should be "useful" in some sense (analogous to correctness property of standard TWO-PC protocols). For example, a TWO-PC protocol (output denoted by $\widehat{h}(x,y)$ for inputs x, y) for computing the Hamming distance $h(x,y)$ is (γ,ξ)-additive-useful [25,21,6] if and only if $\Pr[|h(x) - \widehat{h}(x)| > \gamma(\kappa)] \leq \xi(\kappa)$. We define and work with a somewhat more general notion, (s,ξ)-usefulness with respect to a predicate P, which can be found in the full version. We can now define privacy-preserving TWO-PC.

Definition 7 (Privacy-preserving two-party computation). *An interactive protocol ensemble $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ is (s,ξ) ϵ_κ-*type private two-party computation protocol for $h = (h_f, h_g)$ with respect to P *if for* both f_κ, g_κ, *the ensemble ensures ϵ-*type *and provides (ξ, s)-usefulness for f_κ with respect to predicate P, where* type $\in$ {DP, IND-CDP, SIM-CDP}.

There is rich literature considering notions of security for MPC/TWO-PC simulation-based security [26,27], super-polynomial simulation [28,29], input indistinguishable computation [30], etc. Our notions of private TWO-PC (definition 7) can be seen as new notions of "security" where the only concern for the parties is the privacy of their inputs—here the notion of privacy being (computational) differential privacy. As these notions do not demand efficient simulation (note that even in ϵ_κ-IND-CDP, we do not require the "ideal" F_κ is efficiently computable), they may be easier to achieve; though accuracy may now be the difficult dimension of this aspect.

We note that although our presentation is only for TWO-PC, an extension to MPC is straightforward.

"Ideal/Real" Style Definition of Privacy: ϵ-SIM$^+$-CDP. We now present a new definition, ϵ_κ-SIM$^+$-CDP, which is of particular interest in the context of interactive TWO-PC (for the non-interactive case, it reduces to ϵ_κ-SIM-CDP). This definition is inspired from the "ideal/real" paradigm style definitions used for defining secure TWO-PC/MPC (see [31,32]).

Let P_1, P_2 be two parties, with private inputs a, b respectively, who would like to compute a function $h(a, b)$. What would be the "ideal" situation for the two parties? If there were a trusted third party $\mathcal{T}$ available, P_1, P_2 could first fix a *ϵ-differentially-private* mechanism $\widehat{h}$ that would be "useful" for approximating h according to some metric, and then hand over their inputs a, b to $\mathcal{T}$, who could then compute $\widehat{h}(a, b)$ with uniformly chosen randomness and provide the output to both the parties. Clearly, this informally described "ideal process" (which is literally known as IDEAL world, in secure MPC literature) provides ϵ-DP. Thus, if we had a secure TWO-PC protocol π that emulates this ideal world for all PPT adversaries, intuitively π would "look differentially private" to these adversaries. Moreover, since $\widehat{h}$ is (information-theoretically) differentially private, we can use any π proven "secure" by simulation and privacy is intuitively maintained even if the simulation is not efficient.

We now present the formal definition. In what follows, we assume familiarity with "ideal/real"-paradigm. For a complete and formal description of IDEAL and REAL experiments, we refer the reader to standard texts (e.g., [31,32]). Our definition differs from the standard definition, solely in the sense that the simulator is not necessarily efficient. Though clear from the definitions, we point out that we are working with the *static* corruption model. To maintain consistency with our notation, our definitions are described via interactive function ensembles.

Definition 8 (SIM$^+$-CDP private two-party computation). *An interactive protocol ensemble* $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ *is* (s, ξ) ϵ_κ-SIM$^+$-CDP private two-party computation protocol *for* $h = (h_f, h_g)$ *with respect to the predicate* P *if there exists an* ϵ_κ-DP *randomized mechanism* $\widehat{h} = (\widehat{h}_f, \widehat{h}_g)$ *such that*

- *Mechanism* $\widehat{h}$ *provides* (s, ξ)*-usefulness for* h *with respect to the predicate* P.
- *The protocol ensemble* $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ *is a secure two-party computation protocol ensemble for the randomized functionality* $\widehat{h}$ *as per the "ideal/real"-style definition of secure two party computation. (see full version)*

Clearly, the definitions of SIM-CDP and SIM$^+$-CDP are similar in asserting the existence of simulators whose output is computationally indistinguishable from the real world's transcripts. The difference between the definitions is that in the former the simulator is restricted to being differentially private but otherwise has unfettered access to the input, while the latter only has access to a differentially-private output, from which it has to reconstruct the entire view. The proofs of the following two theorems are provided in the full version.

Theorem 4

1. SIM$^+$-CDP $\Rightarrow$ SIM-CDP If a protocol ensemble $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ satisfies definition 8, then it also satisfies definition 7 for *type*=SIM-CDP.

2. SIM-CDP $\not\Rightarrow$-SIM$^+$-CDP] There exists a protocol ensemble $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ and a function $h(\cdot,\cdot)$ such that the protocol ensemble $\{\langle f_\kappa(\cdot), g_\kappa(\cdot)\rangle\}_{\kappa\in\mathbb{N}}$ is a

$(0,0)$-additive-useful private two-party computation protocol for h (computing h exactly), that provides ϵ_κ-SIM-CDP but not ϵ_κ-SIM$^+$-CDP.

Protocols: Private two-party computation of the Hamming distance. We now demonstrate the usefulness of our definitions by constructing a *simple* and *efficient* protocol which allows two parties to compute the Hamming distance between their respective inputs in just *two* rounds. This protocol will demonstrate the flexibility that comes with our definitions for designing "privacy" protocols. Note that our "privacy" definitions makes the problem quite different from the work on private set intersection protocols (see, for example [33] and the references therein).

For vectors $a, b \in \{0,1\}^n = \mathcal{D}$ define the *Hamming distance*, denoted $h(a,b)$, to be the number of positions in which a, b differ (equivalently, the vectors can be associated with subsets of an n-element universe). Then using *additive homomorphic encryption* and the transformation by [25], we can construct a protocol private TWO-PC for approximating $h(a,b)$ more efficient than generic constructions. Due to space constraints, this protocol π_h (semi-honest and malicious versions) along with the proof of following theorem, is presented in full version.

Theorem 5. *For every $0 < \epsilon < 1$ there exists a $\gamma \in O(1)$ and a constant ξ (depending only on ϵ, γ) such that for every $a, b \in \{0,1\}^n$ it holds that the output $\widehat{h}$ of the protocol π_h satisfies the following*

$$\Pr[|h^* - \widehat{h}| \geq \gamma] \leq \xi,$$

where $h^* = h(a,b)$ and the probability is taken over the randomness of π_h. Further, π_h provides ϵ_κ-SIM-CDP.

Note that NO protocol which satisfies ϵ-DP with above accuracy guarantee on additive error is known so far. For small *multiplicative* error, however, in the full version of this paper, we present a protocol, π_h^*, which ensures ϵ-DP. We also prove the following theorem there:

Theorem 6. *For every $0 < \epsilon, \gamma, \eta < 1$, there exists $\delta \in \Theta(1 - e^{-\gamma/2})$ such that for every $a, b \in \{0,1\}^n$ satisfying $h(a,b) \in \omega\left(\frac{1}{\epsilon^2\delta^2}\left(\ln(5n/\eta)\right)^4\right)$, it holds that the output $\widehat{h}$ of the protocol π_h^* satisfies the following*

$$\Pr[h^* \leq \widehat{h} \leq (1+\gamma)h^*] \geq 1 - \eta,$$

where $h^* = h(a,b)$ and the probability is taken over the randomness of π_h^*. Further, π_h^* provides ϵ-DP.

The high level idea behind π_h^* is to construct a differentially-private version of the communication-efficient KOR algorithm based on sketches [34]. This is done by properly sanitizing the sketches using a standard randomized response mechanism, which has additive error $\Theta(\sqrt{n})$. The two error sources (intrinsic to KOR and due to randomized response) cannot vanish simultaneously, restricting the minimal value of $h(a, b)$ for which π_h^*'s usefulness guarantee hold (its privacy guarantees are always preserved).

We clarify that although differentially-private protocols compute an approximation to the actual function h, they should not be confused with the *fundamentally different* line of research on "secure approximations" (introduced by Feigenbaum et al. [35]). Due to space constraints, this discussion, along with future research directions is deferred to the full version.

Acknowledgements. We would like to thank Cynthia Dwork, Thomas Holenstein, Frank McSherry, Kunal Talwar, and Adam Smith for inspiring discussions and the anonymous reviewers for helpful comments.

References

1. Reingold, O., Trevisan, L., Tulsiani, M., Vadhan, S.: Dense subsets of pseudorandom sets. In: FOCS 2008 (2008)
2. Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
3. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
4. Dwork, C., Nissim, K.: Privacy-preserving datamining on vertically partitioned databases. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 528–544. Springer, Heidelberg (2004)
5. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In: Principles of Database Systems 2007, pp. 273–282 (2007)
6. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: STOC 2008, pp. 609–618 (2008)
7. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS, pp. 94–103. IEEE Computer Society, Los Alamitos (2007)
8. Feldman, D., Fiat, A., Kaplan, H., Nissim, K.: Private coresets. In: STOC (to appear, 2009)
9. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
10. Green, B., Tao, T.: The primes contain arbitrarily long arithmetic progressions. pre-print arXiv:math/0404188 [math.NT] (April 2004)
11. Tao, T., Ziegler, T.: The primes contain arbitrarily long polynomial progressions. pre-print arXiv:math/0404188 [math.NT] (October 2006)
12. Barak, B., Shaltiel, R., Wigderson, A.: Computational analogues of entropy. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 200–215. Springer, Heidelberg (2003)
13. Reingold, O., Vadhan, S.: Personal Communication
14. Agrawal, R., Evfimievski, A.V., Srikant, R.: Information sharing across private databases. In: ACM SIGMOD Conference, pp. 86–97 (2003)
15. Wright, R.N., Yang, Z.: Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In: KDD, pp. 713–718 (2004)

16. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2004)
17. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
18. Kissner, L., Song, D.X.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
19. McSherry, F.: Privacy integrated queries. In: ACM SIGMOD 2009 (2009)
20. Reingold, O., Trevisan, L., Tulsiani, M., Vadhan, S.: Dense subsets of pseudorandom sets. In: Electronic Colloquium on Computational Complexity (ECCC) (2008)
21. Beimel, A., Nissim, K., Omri, E.: Distributed private data analysis: Simultaneously solving how and what. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 451–468. Springer, Heidelberg (2008)
22. Chor, B., Kushilevitz, E.: A zero-one law for boolean privacy. SIAM J. Discrete Math. 4(1), 36–47 (1991)
23. Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. In: STOC, pp. 59–68. ACM, New York (1986)
24. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: STOC, pp. 75–84 (2007)
25. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
26. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: FOCS 1982, pp. 160–164 (1982)
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC 1987, pp. 218–229 (1987)
28. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
29. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: STOC 2004, pp. 242–251 (2004)
30. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: FOCS 2006, pp. 367–378 (2006)
31. Goldreich, O.: Secure Multiparty Computation (1998) (manuscript, Preliminary Version), http://www.wisdom.weizmann.ac.il/~oded/pp.html
32. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)
33. Camenisch, J., Zaverucha, G.M.: Private intersection of certified sets. In: Financial Cryptography and Data Security (to appear, 2009)
34. Kushilevitz, E., Ostrovsky, R., Rabani, Y.: Efficient search for approximate nearest neighbor in high dimensional spaces. In: STOC 1998, pp. 614–623 (1998)
35. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M., Wright, R.N.: Secure multiparty computation of approximations. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 927–938. Springer, Heidelberg (2001)

Probabilistically Checkable Arguments

Yael Tauman Kalai[1] and Ran Raz[2]

[1] Microsoft Research
[2] Weizmann Institute

Abstract. We give a general reduction that converts any public-coin interactive proof into a one-round (two-message) argument. The reduction relies on a method proposed by Aiello *et al.* [1], of using a Private-Information-Retrieval (PIR) scheme to collapse rounds in interactive protocols. For example, the reduction implies that for any security parameter t, the membership in any language in PSPACE can be proved by a one-round (two-message) argument of size $\text{poly}(n, t)$, which is sound for malicious provers of size 2^t. (Note that the honest prover in this construction runs in exponential time, since she has to prove membership in PSPACE, but we can choose t such that 2^t is significantly larger than the running time of the honest prover).

A *probabilistically checkable argument* (PCA) is a relaxation of the notion of probabilistically checkable proof (PCP). It is defined analogously to PCP, except that the soundness property is required to hold only *computationally.* We consider the model where the argument is of one round (two-message), where the verifier's message depends only on his (private) randomness. We show that for membership in many NP languages, there are PCAs (with efficient honest provers) that are of size polynomial in the size of the *witness.* This compares to the best PCPs that are of size polynomial in the size of the *instance* (that may be significantly larger). The number of queries to these PCAs is poly-logarithmic.

The soundness property, in all our results, relies on exponential hardness assumptions for PIR schemes.

1 Introduction

In this paper, we define and study the new notion of *probabilistically checkable argument* (PCA), (see Subsection 1.1). We give a general reduction that uses a *poly-logarithmic* PIR scheme to convert any public-coin interactive proof into a one-round (two-message) argument, (see Subsection 1.2). For example, the reduction shows that membership in any language in PSPACE can be proved by a one-round (two-message) argument of polynomial size, (see Subsection 1.3). Similarly, we give a general reduction that converts any efficient *interactive*-PCP, with certain properties, into a short PCA, (see Subsections 1.4, 1.5). Combined with a recent efficient construction of interactive-PCPs [14], the reduction gives, for membership in many NP languages, PCAs that are significantly shorter than the known PCPs for these languages, (see Subsection 1.6).

1.1 Probabilistically Checkable Arguments

The PCP theorem states that the satisfiability of a formula $\Phi(z_1,\ldots,z_k)$ of size n can be proved by a proof of size poly(n) that can be verified by reading only a constant number of its bits [5,10,3,2]. Note, however, that in many cases the size of the witness, k, is significantly smaller than the size of the instance, n. A central line of research in the area of PCPs is devoted to constructing short PCPs. In particular, one could hope that the satisfiability of a formula $\Phi(z_1,\ldots,z_k)$ of size n could be proved by PCPs of size poly(k), rather than poly(n), (see for example [15]). However, a very interesting recent result of Fortnow and Santhanam shows that this is very unlikely, as it implies that $NP \subseteq coNP/poly$ [12].

In this paper, we consider the relaxed setting of *probabilistically checkable argument* (PCA). We show that for many NP languages, there are PCAs that are of size polynomial in the size of the witness, rather than polynomial in the size of the instance. The number of queries to these PCAs is poly-logarithmic in n.

Roughly speaking, a PCA is a relaxation of the notion of PCP, where the soundness property is required to hold only *computationally*. We consider the model where the argument is of one round (two-message), where the verifier's message depends only on his (private) randomness[1]. Before the protocol starts, the verifier generates two strings that we refer to below as a "secret key" and a "public key", and sends the public key in the first message. The prover's message depends on the verifier's public key. The verifier in turn will need to use his secret key for verification.[2]

More precisely, a PCA system is associated with three algorithms: a *key generation algorithm* $\mathcal{G}$, a *proof generation algorithm* $\mathcal{P}$, and a *verification algorithm* $\mathcal{V}$. It is also associated with five parameters t,p,q,c,s, where t,p,q are integers and c,s are reals, s.t. $0 \le s < c \le 1$. (Informally, t is the *security parameter*, p is the *size* of the PCA, q is the *number of queries* allowed to the PCA, c is the *completeness* parameter and s is the *soundness* parameter). We think of the parameters t,p,q,c,s as functions of the instance size n.

Let L be an NP language, defined by $L = \{x : \exists w \;\; s.t. \;\; (x,w) \in R_L\}$. Suppose that Alice wishes to prove to Bob that $x \in L$. Assume that Bob applied in the past the key generation algorithm $\mathcal{G}$, and thus is associated with a pair of secret and public keys $(SK, PK) \leftarrow \mathcal{G}(1^t)$. Assume that Bob sent to Alice the public key PK. We assume that both Alice and Bob know L and that they both get as input an instance x of size n. Alice gets an additional input w (supposedly a witness for the membership of $x \in L$). A PCA system allows Alice to generate

[1] We note that one could consider several other models for PCA, such as, PCA in the common random string model, where there is a public random string that both the prover and the verifier can see, and the argument is composed of only one message sent by the prover to the verifier.

[2] We note that each pair of secret and public keys can only be used once. Our soundness' proof are only valid in this case. Moreover, it was noted to us by Rafael Pass that by observing whether a verifier accepted or rejected a contrived message (sent by the prover), the prover can gain information about the secret key of the verifier.

a string $\pi \leftarrow \mathcal{P}(x, w, PK)$ of p bits. Bob is allowed to access at most q bits of the string π, and based on these bits he decides whether to accept or reject the statement $x \in L$. We require the following completeness and soundness properties:

1. **Completeness:** For any $x \in L$ and any witness w (given to the prover as input) such that $(x, w) \in R_L$, the verifier, associated with a pair of secret and public keys $(SK, PK) \leftarrow \mathcal{G}(1^t)$, accepts $\pi \leftarrow \mathcal{P}(x, w, PK)$ with probability at least c. Namely,
$$\Pr[\mathcal{V}^{\pi}(x, SK, PK) = 1] \geq c$$
where the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^t)$, over $\pi \leftarrow \mathcal{P}(x, w, PK)$, and over the randomness of $\mathcal{V}$.
2. **Soundness:** For any $x \notin L$ and any cheating prover $\tilde{\mathcal{P}}$ of size $\leq 2^t$,
$$\Pr[\mathcal{V}^{\tilde{\pi}}(x, SK, PK) = 1] \leq s$$
where $\tilde{\pi} = \tilde{\mathcal{P}}(PK)$, and the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^t)$ and over the randomness of $\mathcal{V}$.

For the formal definition of PCA, see Section 2.

1.2 From Interactive Proofs to One-Round Arguments

We propose a general method for reducing the number of rounds in any public-coin interactive proof (that is, an interactive proof where all the bits sent by the verifier are random, and consist of the verifier's random coin tosses). More specifically, our method uses a PIR scheme to convert any public-coin interactive proof into a one-round (two-message) argument. The idea of using a PIR scheme to reduce the round complexity in interactive protocols was proposed by Aiello *et al.* [1], who used a PIR scheme to convert the (short) 4-message argument for NP proposed by [19,21], into a (short) 2-message protocol. Although their 2-message protocol is very natural, attempts to prove its soundness have failed. Moreover, Dwork *et al.* [9] exhibit inherent difficulties in such attempts (for the protocol of [1] and for extensions of this protocol). Dwork *et al.* note that the essence of the problem is that seemingly independant executions of PIR schemes may have, so called, *spooky interactions*. We prove that our method, which is based on the ideas of Aiello *et al.* [1], is sound, if the initial interactive protocol involves only one prover, and is a *proof.*

A *Private Information Retrieval* (PIR) scheme, a concept introduced by Chor, Goldreich, Kushilevitz, and Sudan [7] and by Kushilevitz and Ostrovsky [16][3], allows a user to retrieve information from a database in a private manner. More formally, the database is modeled as an N bit string $x = (x_1, \ldots, x_N)$, out of

[3] The original PIR scheme of [7] had information theoretic privacy but required several copies of the database that cannot interact with each other. The first PIR scheme with a single database (with privacy under computational assumptions) was obtained in [16].

which the user retrieves the i'th bit x_i, without revealing any information about the index i. A trivial PIR scheme consists of sending the entire database to the user, thus satisfying the PIR privacy requirement in the information-theoretic sense. A PIR scheme with communication complexity smaller than N is said to be *non-trivial.* In this paper, we are interested in *poly-logarithmic* PIR schemes, formally defined by Cachin *et al.* [6]. Roughly speaking, a poly-logarithmic PIR scheme is a PIR scheme with poly-logarithmic communication complexity. For the formal definition of poly-logarithmic PIR scheme, see Subsection 3.2.

Roughly speaking, we are able to prove the following result. Assume the existence of a poly-logarithmic PIR scheme (as defined in [6]). Assume that there exists a public-coin interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in some language L; with communication complexity ℓ, completeness c, and soundness s. Then for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\text{poly}(t)$, completeness $c - 2^{-t^2}$, and soundness $s + 2^{-t^2}$ against malicious provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\text{poly}(t, n)$ (assuming that $\mathcal{V}$ runs in time $\text{poly}(n)$). The prover $\mathcal{P}'$ runs in time $\text{poly}(T, t, 2^\lambda)$, where T is the running time of $\mathcal{P}$, and λ is the total number of bits sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive proof system $(\mathcal{P}, \mathcal{V})$.

Moreover, the resulting one-round argument system $(\mathcal{P}', \mathcal{V}')$ has the property that the first message, sent by $\mathcal{V}'$, depends only on the random coin tosses of $\mathcal{V}'$ (and is independent of the instance x), and can be computed in time $\text{poly}(t)$.

The main idea of the proof is as follows. For every round i of the protocol $(\mathcal{P}, \mathcal{V})$, the prover $\mathcal{P}'$ prepares a database DB_i (of size at most 2^λ) of the response of $\mathcal{P}$ (in round i) on all the possibilities of bits sent by $\mathcal{V}$ (in all rounds). The verifier $\mathcal{V}'$ retrieves the response of $\mathcal{P}$ from this database, using a PIR scheme. This is done simoulataneously for all rounds of the protocol. Intuitively, the use of a PIR scheme ensures that when the prover $\mathcal{P}'$ prepares the database DB_i, she cannot use information about the bits sent by the verifier in later rounds. As mentioned above, Dwork *et al.* show that this intuition is misleading in many cases [9]. Nevertheless, we are able to prove that our protocol is sound. Thus, in this particular case, spooky interactions are not a problem.

Note that the running time of the honest prover $\mathcal{P}'$ is exponential in λ, where λ is the total number of bits sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive proof system $(\mathcal{P}, \mathcal{V})$. This is the case because $\mathcal{P}'$ has to handle the databases DB_i of size 2^λ. We are able to improve a little bit over that and to prove a similar result, where the prover's running time is $\text{poly}(T, t, 2^\lambda)$, where this time λ is the maximum number of bits of the verifier's messages that the prover "needs" to read in order to compute one bit to be sent to the verifier.

For more details, see Section 4.

The problem of reducing the number of rounds in interactive protocols was previously studied in a large number of works. The most famous heuristic for reducing the number of rounds in general protocols is the Fiat-Shamir method [11]. It was shown in [4,13] that in certain cases where the original protocol is an *argument* the obtained protocol is not sound. It is still not known whether or not the obtained protocol is sound if the original protocol is a *proof.*

A different approach, more related to ours, to reduce the number of rounds in interactive proofs was previously done by Damgard, Fazio and Nicolosi [8]. Damgard, Fazio and Nicolosi used a homomorphic encryption, rather than PIR, and their results only apply for 3 messages protocols that have the additional property that the third message is linear in the second message. The proof of soundness of their protocols is obtained by a similar approach to the one taken here.

1.3 One-Round Arguments for PSPACE

The reduction from Subsection 1.2 can be used to convert any public-coin interactive proof into a one-round argument. In particular, it can be used for giving one-round arguments for membership in PSPACE languages.

For any language L in PSPACE, there exists a public-coin interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in L; with communication complexity $\ell = \text{poly}(n)$, completeness 1, and exponentially small soundness [20,22]. Using our general reduction, we can translate this interactive proof into a one-round argument as follows: For any security parameter $t \geq \max\{\ell, n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\text{poly}(t)$, completeness $1 - 2^{-t}$, and soundness 2^{-t} against malicious provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\text{poly}(t)$. The prover $\mathcal{P}'$ runs in time $\text{poly}(t, 2^\ell)$.

Note that the running time of the honest prover $\mathcal{P}'$ is exponential. This seems necessary because the prover has to prove membership in PSPACE languages. Note, however, that we can choose t to be significantly larger than ℓ, say, $t = \ell^3$. In this case, the honest prover runs in time $\text{poly}(2^\ell)$, while the proof is sound for malicious provers of size $\leq 2^{\ell^3}$.

1.4 Interactive-PCP

An interactive-PCP (say, for the membership $x \in L$) is a combination of a PCP and a short interactive proof. Roughly speaking, an interactive-PCP is a proof that can be verified by reading only a small number of its bits, with the help of a short interactive proof.

More precisely, let L be an NP language, defined by $L = \{x : \exists w \;\; s.t. \;\; (x, w) \in R_L\}$. Let p, q, l, c, s be parameters as follows: p, q, l are integers and c, s are reals, s.t. $0 \leq s < c \leq 1$. (Informally, p is the *size of the PCP*, q is the *number of queries* allowed to the PCP, l is the *communication complexity* of the interactive proof, c is the *completeness* parameter and s is the *soundness* parameter). We think of the parameters p, q, l, c, s as functions of the instance size n. An interactive-PCP with parameters (p, q, l, c, s) for membership in L is an interactive protocol between an (efficient) prover P and an (efficient) verifier V, as follows:

We assume that both the prover and the verifier know L and get as input an instance x of size n, and the prover gets an additional input w (supposed to be a witness for the membership $x \in L$). In the first round of the protocol, the prover generates a string π of p bits. (We think of π as an encoding of the witness w). The verifier is still not allowed to access π. The prover and the verifier then apply an interactive protocol, where the total number of bits communicated is l. During the protocol, the verifier is allowed to access at most q bits of the string π. After the interaction, the verifier decides whether to accept or reject the statement $x \in L$. We require the following completeness and soundness properties:

There exists an (efficient) verifier V such that:

1. Completeness: There exists an (efficient) prover P, such that: for every $x \in L$ and any witness w (given to the prover P as an input), if $(x, w) \in R_L$ then the verifier accepts with probability at least c.
2. Soundness: For any $x \notin L$ and any (not necessarily efficient) prover $\tilde{P}$, and any w (given to the prover $\tilde{P}$ as an input), the verifier accepts with probability at most s.

For the formal definition of interactive-PCP, see Subsection 3.1.

1.5 From Interactive-PCP to PCA

We give a general reduction that converts any efficient interactive-PCP, with certain properties, into a short PCA. The main idea is to use the reduction from Subsection 1.2 to convert the interactive phase of the interactive-PCP into a one-round argument.

Roughly speaking, we are able to prove the following result. Assume the existence of a poly-logarithmic PIR scheme (as defined in [6]). Assume that there exists an interactive-PCP system $(\mathcal{P}, \mathcal{V})$ with parameters p, q, ℓ, c, s for some NP language L, such that the interactive phase of $(\mathcal{P}, \mathcal{V})$ is public-coin, and each bit sent by the prover in the interactive phase depends on at most λ bits sent by the verifier. Then, for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a PCA system $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ with parameters t, p', q', c', s' for the language L, where $p' = \text{poly}(p, t)$, $q' = \text{poly}(q, t)$, $c' \geq c - 2^{-t^2}$, and $s' \leq s + 2^{-t^2}$. The prover $\mathcal{P}'$ runs in time $\text{poly}(t, n, 2^{\lambda})$.

For more details, see Section 5.

1.6 Short PCAs for Satisfiability

Efficient constructions of interactive-PCPs were given in [18,14]. In particular, in [14], the following theorem was proven.

Let $\Phi(z_1, \ldots, z_k)$ be a Boolean circuit of size n and depth d, and assume without loss of generality that $k \geq \log n$ (otherwise, it is easy to check the satisfiability of Φ in time $\text{poly}(n)$). Let s be such that, $\log n \leq s \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an interactive-PCP with the following parameters. Size of the PCP: $p = \text{poly}(k, d)$. Number of queries: $q = \text{poly}(s)$. Communication complexity of the interactive phase: $\ell = \text{poly}(d, s)$. Completeness: 1. Soundness: 2^{-s}. Moreover, the interactive phase is public-coin, and each message sent by the prover depends only on the preceding $\lambda = O(\log n)$ bits sent by the verifier.

Using our reduction from interactive-PCP to PCA, together with the theorem of [14], one obtains the following result. (Our reduction and the theorem of [14] were obtained roughly at the same time. The result below follows from their combination).

Let $\Phi(z_1, \ldots, z_k)$ be a Boolean circuit of size n and depth d. Let t be a security parameter, such that, $\log n \leq t \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an efficient PCA system (i.e., PCA with an efficient prover), with the following parameters. Size of the PCA: $p = \text{poly}(k, d, t)$. Number of queries: $q = \text{poly}(d, t)$. Completeness: $1 - 2^{-t}$. Soundness: 2^{-t}.

In particular, if $\Phi(z_1, \ldots, z_k)$ is a Boolean formula of size n, and $\log n \leq t \leq \text{poly}(n)$, the satisfiability of Φ can be proved by an efficient PCA system, with the following parameters. Size of the PCA: $p = \text{poly}(k, t)$. Number of queries: $q = \text{poly}(t)$. Completeness: $1 - 2^{-t}$. Soundness: 2^{-t}.

2 Definition of PCA

Let L be any NP language defined by $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$. Let t, p, q, c, s be parameters that satisfy the following: The parameters $t, p, q : \mathbb{N} \rightarrow \mathbb{N}$ are integers, and the parameters $c, s : \mathbb{N} \rightarrow [0, 1]$ are reals, such that for every $n \in \mathbb{N}$, $0 \leq s(n) < c(n) \leq 1$.

Definition 1. *A triplet $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ of probabilistic Turing machines is a* **PCA system** *for L with parameters (t, p, q, c, s), if the following holds:*

- *$\mathcal{G}$ is a probabilistic Turing machine that runs in time $\text{poly}(t)$, and $\mathcal{V}$ is a probabilistic oracle machine that runs in time $\text{poly}(t, n)$.*
- *For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$) and every $(SK, PK) \leftarrow \mathcal{G}(1^{t(n)})$, the algorithm $\mathcal{P}(x, w, PK)$ generates a bit string π of size at most $p(n)$, and the oracle machine $\mathcal{V}^{\pi}(x, SK, PK)$ reads at most $q(n)$ bits of π.*
- **Completeness:** *For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$),*
$$\Pr[\mathcal{V}^{\pi}(x, SK, PK) = 1] \geq c(n)$$
(where the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^{t(n)})$, over $\pi \leftarrow \mathcal{P}(x, w, PK)$, and over the randomness of $\mathcal{V}$).
- **Soundness:** *For every $x \notin L$ (where $|x| = n$), and every cheating prover $\tilde{\mathcal{P}}$ of size $\leq 2^{t(n)}$,*
$$\Pr[\mathcal{V}^{\tilde{\pi}}(x, SK, PK) = 1] \leq s(n)$$
(where $\tilde{\pi} = \tilde{\mathcal{P}}(PK)$, and the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^{t(n)})$ and over the randomness of $\mathcal{V}$).

Remark. Note that in Definition 1 we did not specify the complexity of $\mathcal{P}$. We say that a PCA system $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is *efficient* if $\mathcal{P}$ runs in time $\text{poly}(t, n)$.

3 Preliminaries

3.1 Interactive-PCP (IPCP)

Let L be any NP language defined by $L = \{x : \exists w \text{ s.t. } (x,w) \in \mathcal{R}_L\}$. Let p,q,ℓ,c,s be parameters that satisfy the following: The parameters $p,q,\ell : \mathbb{N} \to \mathbb{N}$ are integers, and the parameters $c,s : \mathbb{N} \to [0,1]$ are reals, such that for every $n \in \mathbb{N}$, $0 \leq s(n) < c(n) \leq 1$.

Definition 2. *A pair $(\mathcal{P},\mathcal{V})$ of probabilistic polynomial time interactive Turing machines is an* interactive-PCP *for L with parameters (p,q,ℓ,c,s), if for every $(x,w) \in \mathcal{R}_L$ the prover $\mathcal{P}(x,w)$ generates a bit string π of size at most $p(n)$ (where $n = |x|$), such that the following properties are satisfied.*

– **Completeness:** *For every $(x,w) \in \mathcal{R}_L$,*

$$\Pr[(\mathcal{P}(x,w),\mathcal{V}^{\pi}(x)) = 1] \geq c(n)$$

(where $n = |x|$, and the probability is over the random coin tosses of $\mathcal{P}$ and $\mathcal{V}$).

– **Soundness:** *For every $x \notin L$, every (unbounded) interactive Turing machine $\tilde{\mathcal{P}}$, and every string $\tilde{\pi} \in \{0,1\}^*$,*

$$\Pr[(\tilde{\mathcal{P}}(x),\mathcal{V}^{\tilde{\pi}}(x)) = 1] \leq s(n)$$

(where $n = |x|$, and the probability is over the random coin tosses of $\mathcal{V}$).

– **Complexity:** *The communication complexity of the protocol $(\mathcal{P}(x,w),\mathcal{V}^{\pi}(x))$ is at most $\ell(n)$, and $\mathcal{V}$ reads at most $q(n)$ bits of π.*

3.2 Private Information Retrieval (PIR)

A PIR scheme consists of three algorithms: Q^{PIR}, D^{PIR} and R^{PIR}. The query algorithm Q^{PIR} takes as input a security parameter t, the database size N, and an index $i \in [N]$ (that the user wishes to retrieve from the database). It outputs a query q, which should reveal no information about the index i, together with an additional output s, which is kept secret by the user and will later assist the user in retrieving the desired element from the database. The database algorithm D^{PIR} takes as input a security parameter t, the database $(x_1,\ldots,x_N)$ and a query q, and outputs an answer a. This answer enables the user to retrieve x_i, by applying the retrieval algorithm R^{PIR}, which takes as input a security parameter t, the database size N, an index $i \in [N]$, a corresponding pair (q,s) obtained from the query algorithm, and the database answer a corresponding to the query q. It outputs a value which is supposed to be the i'th value of the database.

In this paper we are interested in *poly-logarithmic* PIR schemes, formally defined by Cachin *et al.* [6], as follows.[4]

[4] Definition 3 is not worded exactly as the one in [6], but was shown to be equivalent to it in [17].

Definition 3. *Let t be the security parameter and N be the database size. Let Q^{PIR} and D^{PIR} be probabilistic circuits, and let R^{PIR} be a deterministic circuit. We say that $(Q^{PIR}, D^{PIR}, R^{PIR})$ is a* poly-logarithmic private information retrieval scheme *if the following conditions are satisfied:*

1. (Size Restriction:) *Q^{PIR} and R^{PIR} are of size $\leq \text{poly}(t, \log N)$, and D^{PIR} is of size $\leq \text{poly}(t, N)$. The output of Q^{PIR} and D^{PIR} is of size $\leq \text{poly}(t, \log N)$.*
2. (Correctness:) *$\forall N$, $\forall t$, $\forall$database $x = (x_1, \ldots, x_N) \in \{0,1\}^N$, and $\forall i \in [N]$,*

$$\Pr[R^{PIR}(t, N, i, (q, s), a) = x_i \mid (q, s) \leftarrow Q^{PIR}(t, N, i), a \leftarrow D^{PIR}(t, x, q)]$$

$$\geq 1 - 2^{-t^3}.$$

3. (User Privacy:) *$\forall N$, $\forall t$, $\forall i, j \in [N]$, and $\forall$adversary $\mathcal{A}$ of size at most 2^{t^3},*

$$\big|\Pr[\mathcal{A}(t, N, q) = 1 \mid (q, s) \leftarrow Q^{PIR}(t, N, i)] -$$
$$\Pr[\mathcal{A}(t, N, q) = 1 \mid (q, s) \leftarrow Q^{PIR}(t, N, j)]\big| \leq 2^{-t^3}.$$

4 From Interactive Proofs to One-Round Arguments

In this section, we propose a general method for reducing the number of rounds in any public-coin interactive proof (that is, an interactive proof where all the bits sent by the verifier are random, and consist of the verifier's random coin tosses). More specifically, our method uses a PIR scheme to convert any public-coin interactive proof into a one-round (two-message) argument.

Lemma 1. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3).*[5] *Assume that there exists a public-coin interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in some language L, with communication complexity ℓ, completeness c, and soundness s. Then for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\ell' = \text{poly}(t, \ell) = \text{poly}(t)$, completeness $c' \geq c - 2^{-t^2}$, and soundness $s' \leq s + 2^{-t^2}$ against provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\leq \text{poly}(t, n)$ (assuming that $\mathcal{V}$ runs in time $\text{poly}(n)$). The prover $\mathcal{P}'$ runs in time $\leq \text{poly}(T, t, 2^\lambda)$, where T is the running time of $\mathcal{P}$, and λ is the total number of bits sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive proof system $(\mathcal{P}, \mathcal{V})$.*

Moreover, the resulting one-round argument system $(\mathcal{P}', \mathcal{V}')$ has the property that the first message, sent by $\mathcal{V}'$, depends only on the random coin tosses of $\mathcal{V}'$ (and is independent of the instance x),[6] *and can be computed in time $\leq \text{poly}(t)$.*

[5] We assume the existence of such a PIR scheme for any parameters N and t.

[6] The fact that this message depends *only* on the random coin tosses of $\mathcal{V}'$ (and is independent of x) will be crucial when converting an interactive-PCP system into a PCA system in Section 5.

Rather than proving Lemma 1 directly, we prove a stronger and more general lemma. The more general lemma converts any interactive proof $(\mathcal{P},\mathcal{V})$, in which the verifier's messages depend only on the verifier's random coin tosses (and in particular, any public-coin interactive proof), into a one-round argument $(\mathcal{P}',\mathcal{V}')$. Moreover, the resulting prover $\mathcal{P}'$ is more efficient than the one in the statement of Lemma 1. More specifically, the running time of $\mathcal{P}'$ is $\leq \text{poly}(T,t,2^{\lambda})$, where T,t are as in the statement of Lemma 1 (i.e., T is the running time of $\mathcal{P}$, and t is the security parameter of the underlying PIR scheme), but λ here is defined differently than in Lemma 1. Recall that in Lemma 1, λ was the *total* number of bits sent by the verifier. Intuitively, here λ is the maximum number of bits of the verifier's messages, that the prover "needs" to read in order to compute one bit to be sent to the verifier. Namely, if λ_i denotes the number of bits of the verifier's messages, that the prover "needs" to read in order to compute its i'th bit (to be sent to the verifier), then $\lambda = \max\{\lambda_1,\ldots,\lambda_\ell\}$. We formalize this via the following definition of a *history-ignorant* interactive proof.

Definition 4. *An ℓ-round interactive proof $(\mathcal{P},\mathcal{V})$ for proving membership in a language L is said to be* history-ignorant*, if for every input $x \in L$, every auxiliary input $w \in \{0,1\}^*$,*[7] *and for every $i \in [\ell]$, the message sent by the (honest) prover $\mathcal{P}(x,w)$ in the i'th round of the protocol $(\mathcal{P}(x,w),\mathcal{V}(x))$ depends only on the message sent by $\mathcal{V}$ in the i'th round of the protocol (and on x,w and the random coin tosses of $\mathcal{P}$),*[8] *and does not depend on the messages sent by $\mathcal{V}$ before the i'th round.*

In the generalized lemma we propose a method for converting any ℓ-round *history-ignorant* interactive proof $(\mathcal{P},\mathcal{V})$ (where the verifier's messages depend only on the verifier's random coin tosses) into a one-round argument $(\mathcal{P}',\mathcal{V}')$. As in Lemma 1, we show that the completeness and soundness parameters remain almost the same, and the communication complexity increases by at most a polynomial factor in the security parameter t. However, here the running time of the resulting prover $\mathcal{P}'$ is $\leq \text{poly}(T,t,2^{\lambda})$, where now λ is the length of the longest message sent by $\mathcal{V}$ (rather than the total number of bits sent by $\mathcal{V}$). Namely, if $m_1,\ldots,m_\ell$ are the ℓ messages sent by $\mathcal{V}$ throughout the protocol $(\mathcal{P},\mathcal{V})$ (where the message m_i is sent by $\mathcal{V}$ in round i), then

$$\lambda \overset{\text{def}}{=} \max_{i\in[\ell]}\{|m_i|\}.$$

Lemma 2. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3). Assume that there exists a history-ignorant interactive proof system $(\mathcal{P},\mathcal{V})$ for proving membership in some language L, where the verifier's messages depend only on the verifier's random coin tosses (and are independent of the interaction and the input). Let ℓ be the communication*

[7] As is common, we allow the prover in the interactive proof system to use an auxiliary input, supposedly a witness for $x \in L$.

[8] We think of each round as consisting of a message sent by the verifier $\mathcal{V}$ followed by a message sent by the prover $\mathcal{P}$.

complexity, c be the completeness parameter, and s be the soundness parameter of the proof system $(\mathcal{P}, \mathcal{V})$. Denote by λ the length of the longest message sent by $\mathcal{V}$, and assume that $\mathcal{V}$ uses at most $O(\ell \cdot \lambda)$ random bits. Then for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\ell' = \mathrm{poly}(t, \ell) = \mathrm{poly}(t)$, completeness $c' \geq c - 2^{-t^2}$, and soundness $s' \leq s + 2^{-t^2}$ against provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\leq \mathrm{poly}(t, n)$ (assuming that $\mathcal{V}$ runs in time $\mathrm{poly}(n)$). The prover $\mathcal{P}'$ runs in time $\leq \mathrm{poly}(T, t, 2^\lambda)$, where T is the running time of $\mathcal{P}$.

Moreover, the resulting one-round argument system $(\mathcal{P}', \mathcal{V}')$ has the property that the first message, sent by $\mathcal{V}'$, depends only on the random coin tosses of $\mathcal{V}'$ (and is independent of the instance x). If each message sent by $\mathcal{V}$ can be computed in time $\leq \mathrm{poly}(t)$ then the first message sent by $\mathcal{V}'$ can also be computed in time $\leq \mathrm{poly}(t)$.

Before proving Lemma 2, we show that this lemma can be used to convert any (not necessarily history-ignorant) interactive proof $(\mathcal{P}, \mathcal{V})$, where the verifier's messages depend only on the verifier's random coin tosses, into a one-round argument $(\mathcal{P}', \mathcal{V}')$, where the running time of the resulting prover $\mathcal{P}'$ is $\leq \mathrm{poly}(T, t, 2^\lambda)$, and where λ is the maximum number of bits of the verifier's messages that the underlying prover $\mathcal{P}$ "needs" to read in order to compute one bit to be sent to the verifier. We assume for simplicity that in the protocol $(\mathcal{P}, \mathcal{V})$, the prover $\mathcal{P}$ sends in each round a *single* bit. This is without loss of generality since we can always increase the number of rounds artificially.

The idea is the following:[9] First convert the interactive proof $(\mathcal{P}, \mathcal{V})$ into the following *history-ignorant* interactive proof $(\mathcal{P}'', \mathcal{V}'')$: The verifier $\mathcal{V}''$ will first prepare all the messages to be sent by $\mathcal{V}$ in the protocol $(\mathcal{P}, \mathcal{V})$. Note that this can be done in advance since according to our assumption, the verifier's messages depend only on the verifier's random coin tosses (and are independent of the interaction and the input). Then, in each round i, the verifier $\mathcal{V}''$ will send $\mathcal{P}''$ all the bits that $\mathcal{P}$ "needs" in order to compute its i'th bit (i.e., its i'th round message) in the protocol $(\mathcal{P}, \mathcal{V})$. The prover $\mathcal{P}''$ will emulate the prover $\mathcal{P}$, while reading only the message sent by $\mathcal{V}''$ in the i'th round. This results with a history-ignorant protocol, with the same completeness and soundness parameters, and where the communication complexity increases by at most a polynomial factor. The thing to notice is that in the i'th round the verifier $\mathcal{V}''$ sends a message of size λ_i (where λ_i is the number of bits of the verifier's messages, that the prover $\mathcal{P}$ "needs" to read in order to compute its i'th bit). What remains is to apply Lemma 2 to the history-ignorant protocol $(\mathcal{P}'', \mathcal{V}'')$ in order to obtain the desired one-round argument $(\mathcal{P}', \mathcal{V}')$.

Proof of Lemma 2. (In this version, due to space limitation, we only describe the protocol and do not give the full proof). Fix any history-ignorant interactive proof system $(\mathcal{P}, \mathcal{V})$ as in the statement of the lemma, for proving membership

[9] The discussion in this paragraph is only an intuition. The formal result is given by Definition 4 and Lemma 2.

in some language L. Denote by λ the length of the longest message sent by $\mathcal{V}$ in the interactive proof $(\mathcal{P}, \mathcal{V})$ $(\lambda \leq \ell)$. We assume for simplicity (and without loss of generality) that this protocol consists of exactly ℓ rounds, where in each round $\mathcal{V}$ sends a message of size exactly λ, and $\mathcal{P}$ sends a single bit.

Fix any security parameter $t \geq \max\{\ell, \log n\}$. Let

$$(Q^{PIR}, D^{PIR}, R^{PIR})$$

be a poly-logarithmic PIR scheme, with respect to security parameter t and database size $N \stackrel{\text{def}}{=} 2^{\lambda}$. (We refer the reader to Subsection 3.2 for the definition of a poly-logarithmic PIR scheme.) We next describe how to convert $(\mathcal{P}, \mathcal{V})$ into a one-round argument system $(\mathcal{P}', \mathcal{V}')$ as in the statement of the lemma.

Fix any $x \in \{0,1\}^*$ (supposedly $x \in L$) and any string $w \in \{0,1\}^*$ given to the prover as auxiliary input. The one-round argument $(\mathcal{P}'(x,w), \mathcal{V}'(x))$ proceeds as follows:

– The verifier $\mathcal{V}'(x)$ sends the first message, computed as follows:

  1. Choose a random string $r_v \in_R \{0,1\}^{O(\ell \cdot \lambda)}$, to be used when emulating the underlying verifier $\mathcal{V}(x)$.
  2. Compute the ℓ messages $m_1, \ldots, m_\ell \in \{0,1\}^{\lambda}$ sent by $\mathcal{V}(x)$ (with randomness r_v). Note that these messages can be computed in advance since in the protocol $(\mathcal{P}, \mathcal{V})$ all the messages sent by $\mathcal{V}$ depend only on $\mathcal{V}$'s random coin tosses.
  3. For each $i \in [\ell]$, let $(q_i, s_i) \leftarrow Q^{PIR}(t, N, m_i)$.
  4. Save the values $(r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.[10]
  5. Send $(q_1, \ldots, q_\ell)$ to $\mathcal{P}'$.

  Note that if each message of $\mathcal{V}$ can be computed in time $\leq \text{poly}(t)$ then the message sent by $\mathcal{V}'$ can be computed in time $\leq \text{poly}(t, \ell) = \text{poly}(t)$.

– Upon receiving a message $(q_1, \ldots, q_\ell)$ from $\mathcal{V}'$, the algorithm $\mathcal{P}'(x,w)$ operates as follows:

  1. Choose a random string $r_p \in_R \{0,1\}^T$, to be used when emulating the underlying prover $\mathcal{P}(x,w)$.
  2. For each $i \in [\ell]$, compute an N-size database DB_i as follows: The $m \in \{0,1\}^{\lambda}$ entry of DB_i contains the i'th bit that the prover $\mathcal{P}(x,w)$ (with randomness r_p) would have sent to $\mathcal{V}(x)$, if the i'th-round message sent by $\mathcal{V}(x)$ was m.[11]
  3. For each $i \in [\ell]$, compute $a_i \leftarrow D^{PIR}(t, DB_i, q_i)$.
  4. Send the message $(a_1, \ldots, a_\ell)$ to $\mathcal{V}'$.

[10] We note that the messages $m_1, \ldots, m_\ell$ do not need to be saved since they can be recomputed from r_v. We save them for simplicity.

[11] We are using here the fact that the interactive proof system $(\mathcal{P}, \mathcal{V})$ is history-ignorant.

– Upon receiving the message $(a_1, \ldots, a_\ell)$ from $\mathcal{P}'$, the algorithm $\mathcal{V}'(x)$ operates as follows:

1. Restore the saved values $(r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.
2. For every $i \in [\ell]$, compute $b'_i \overset{\text{def}}{=} R^{PIR}(t, N, m_i, (q_i, s_i), a_i)$.
3. Accept if and only if $\mathcal{V}(x)$, with (initial) randomness r_v, would have accepted the messages $(b'_1, \ldots, b'_\ell)$.[12] ■

5 From Interactive-PCPs to PCAs

In this section, we propose a general method for converting an interactive-PCP system (with certain properties) into a PCA system. This method is very similar to the method of converting an interactive proof into a one-round argument (presented in Section 4). Namely, it uses a PIR scheme to reduce the round complexity of the interactive phase of the interactive-PCP system into one round (two-messages). Then, the first message (sent by the verifier) in this one-round protocol, is interpreted as the verifier's public-key in the PCA system; and the second message (sent by the prover) in the one-round protocol, together with the interactive-PCP oracle, are interpreted as the PCA string.

Theorem 1. *Assume the existence of a* (*uniform*) *poly-logarithmic PIR scheme* (*as defined in Definition 3*). *Assume that there exists an interactive*-PCP *system* $(\mathcal{P}, \mathcal{V})$ *with parameters* (p, q, ℓ, c, s) *for some NP language L, such that the interactive phase is history-ignorant,*[13] *and each message sent by the verifier in this phase depends only on the verifier's random coin tosses* (*and is independent of the interaction, the* PCP *string* π, *and the input* x), *and can be computed in time* $\leq \text{poly}(\ell)$. *Denote by* λ *the length of the longest message sent from* $\mathcal{V}$ *to* $\mathcal{P}$ *in the interactive phase of the interactive*-PCP *system* $(\mathcal{P}, \mathcal{V})$. *Assume that* $\mathcal{V}$ *uses at most* $O(\ell \cdot \lambda)$ *random bits. Then, for any security parameter* $t \geq \max\{\ell, \log n\}$ *there exists a* PCA *system* $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ *with parameters* (t, p', q', c', s') *for the language L, where* $p' = \text{poly}(p, t)$, $q' = \text{poly}(q, t)$, $c' \geq c - 2^{-t^2}$, *and* $s' \leq s + 2^{-t^2}$. *The prover* $\mathcal{P}'$ *runs in time* $\leq \text{poly}(t, n, 2^\lambda)$.

The proof of this theorem is very similar to the proof of Lemma 2.

Proof of Theorem 1. (In this version, due to space limitation, we only describe the protocol and do not give the full proof). Fix an NP language $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$. Let $(\mathcal{P}, \mathcal{V})$ be an interactive-PCP system for L with parameters (p, q, ℓ, c, s), as in the statement of the theorem. Denote by λ the length of the longest message sent by $\mathcal{V}$ in the interactive phase of $(\mathcal{P}, \mathcal{V})$ $(\lambda \leq \ell)$. We assume for simplicity (and without loss of generality) that the interactive phase consists of exactly ℓ rounds. In each round $i \in [\ell]$, $\mathcal{V}$ sends a message m_i of size

[12] This verdict test may require $\mathcal{V}'(x)$ to use additional (fresh) randomness.

[13] A history-ignorant interactive phase is defined in the same way as a history-ignorant interactive proof. See Definition 4.

exactly λ (which depends only on the verifier's random coin tosses), and $\mathcal{P}$ sends a single bit b_i. For every $(x, w) \in \mathcal{R}_L$, the prover $\mathcal{P}(x, w)$ generates a bit string π of size at most $p(n)$ (where $n = |x|$). We assume for simplicity (and without loss of generality) that π is of size exactly $p(n)$.

Fix any security parameter $t \geq \max\{\ell, \log n\}$. Let $(Q^{PIR}, D^{PIR}, R^{PIR})$ be a poly-logarithmic PIR scheme, with respect to security parameter t and database size $N \overset{\text{def}}{=} 2^\lambda$. (We refer the reader to Subsection 3.2 for the definition of a poly-logarithmic PIR scheme.) We next describe how to convert the interactive-PCP system $(\mathcal{P}, \mathcal{V})$ into a PCA system $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ as in the statement of the theorem.

– $\mathcal{G}'(1^{t(n)})$ operates as follows:

  1. Choose a random string $r_v \in_R \{0,1\}^{O(\ell \cdot \lambda)}$ (to be used when emulating the messages sent by the underlying PCP verifier $\mathcal{V}$).
  2. Compute the ℓ messages $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ sent by $\mathcal{V}$ (with randomness r_v) in the interactive phase of $(\mathcal{P}, \mathcal{V})$. Note that these messages can be computed in advance (since all the messages sent by $\mathcal{V}$ depend only on $\mathcal{V}$'s randomness).
  3. For each $i \in [\ell]$, let $(q_i, s_i) \leftarrow Q^{PIR}(t, N, m_i)$.
  4. Let $PK = (q_1, \ldots, q_\ell)$ and let $SK = (r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.[14]
  5. Output the pair (SK, PK).

  Note that $\mathcal{G}'$ runs in time $\leq \text{poly}(t)$, since each message sent by $\mathcal{V}$ can be computed in time $\leq \text{poly}(\ell) \leq \text{poly}(t)$.

– For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$) and every $(SK, PK) \leftarrow \mathcal{G}'(1^{t(n)})$, the algorithm $\mathcal{P}'(x, w, PK)$ operates as follows.

  1. Parse $PK = (q_1, \ldots, q_\ell)$.
  2. Choose a random string $r_p \in_R \{0,1\}^{\text{poly}(n)}$ (to be used when emulating the underlying PCP prover $\mathcal{P}(x, w)$).
  3. Compute the oracle π, as computed by $\mathcal{P}(x, w)$ (with randomness r_p).
  4. For each $i \in [\ell]$, compute an N-size database DB_i as follows: The $m \in \{0,1\}^\lambda$ entry of DB_i contains the bit b_i that the PCP prover $\mathcal{P}(x, w)$ (with randomness r_p) would have sent to $\mathcal{V}(x)$ in the i'th round of the interactive phase of $(\mathcal{P}, \mathcal{V})$, if the i'th-round message sent by $\mathcal{V}(x)$ was m.[15]
  5. For each $i \in [\ell]$, compute $a_i \leftarrow D^{PIR}(t, DB_i, q_i)$.
  6. Output $\pi' \overset{\text{def}}{=} (\pi, a_1, \ldots, a_\ell)$

– For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$), every $(SK, PK) \leftarrow \mathcal{G}'(1^{t(n)})$, and every string π' (supposedly of the form $\pi' = (\pi, a_1, \ldots, a_\ell)$), the oracle machine $(\mathcal{V}')^{\pi'}(x, SK, PK)$ operates as follows:

[14] We note that the messages $m_1, \ldots, m_\ell$ do not need to be part of the secret key since they can be recomputed from r_v. We save them for simplicity.

[15] We are using here the fact that the interactive phase of $(\mathcal{P}, \mathcal{V})$ is history-ignorant.

1. Query the oracle π' at all the coordinates $i > p(n)$. Denote the values obtained by the oracle by $(a_1, \ldots, a_\ell)$.
2. Parse $PK = (q_1, \ldots, q_\ell)$ and $SK = (r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.
3. For every $i \in [\ell]$, compute $b'_i \overset{\text{def}}{=} R^{PIR}(t, N, m_i, (q_i, s_i), a_i)$.
4. Compute the q oracle queries made by $\mathcal{V}(x)$ (with randomness r_v), assuming the ℓ messages that the prover $\mathcal{P}$ sent $\mathcal{V}$ during the interactive phase were $b'_1, \ldots, b'_\ell$. Denote these q queries by $i_1, \ldots, i_q \in [p(n)]$.
5. Query the oracle at the coordinates $i_1, \ldots, i_q$, and obtain q answers, denoted by $\pi_{i_1}, \ldots, \pi_{i_q}$.
6. Output 1 if and only if the verifier $\mathcal{V}^\pi(x)$ (with randomness r_v) outputs 1 after receiving the messages $b'_1, \ldots, b'_\ell$ from the prover $\mathcal{P}$ and receiving the bits $\pi_{i_1}, \ldots, \pi_{i_q}$ from the oracle. ∎

5.1 Corollaries

We next show how Theorem 1, together with the interactive-PCP system constructed in [14], yields an *efficient* PCA system. We use the following theorem from [14].

Theorem 2. *[14] Let $\Phi(z_1, \ldots, z_k)$ be a (fanin 2) Boolean circuit of size n and depth d, and assume without loss of generality that $k \geq \log n$.*[16] *Let s be such that, $\log n \leq s \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an interactive-PCP with the following parameters. Size of the PCP: $p = \text{poly}(k, d)$. Number of queries: $q = \text{poly}(s)$. Communication complexity of the interactive phase: $\ell = \text{poly}(d, s)$. Completeness: 1. Soundness: 2^{-s}.*

Moreover, the interactive phase is public-coin, the verifier uses at most $O(\ell)$ random bits, and each message sent by the prover depends only on the preceding $\lambda = O(\log n)$ bits sent by the verifier.

The following is an immediate corollary of Theorem 1 and Theorem 2.

Corollary 1. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3). Let $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$ be any NP language, and assume that $\mathcal{R}_L$ is given by a Boolean circuit of size $\text{poly}(n)$ and depth d, where $n = |x|$ denotes the instance size. Let $k = |w|$ denote the witness size. Let t be a security parameter, such that, $\log n \leq t \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an efficient* PCA *system (i.e., PCA with a (honest) prover that runs in time $\text{poly}(n)$), with the following parameters. Size of the PCA: $p = \text{poly}(k, d, t)$. Number of queries: $q = \text{poly}(d, t)$. Completeness: $1 - 2^{-t}$. Soundness: 2^{-t}.*

Proof. Fix any NP language $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$, as above. Fix $s = t+1$. Theorem 2 implies that L has an interactive-PCP system with the following parameters: $p = \text{poly}(k, d, \log n)$, $q = \text{poly}(s)$, $\ell = \text{poly}(d, s)$, completeness 1, and soundness 2^{-s}.

[16] Otherwise, it is easy to check the satisfiability of Φ in time $\text{poly}(n)$.

The fact that each message sent by the (interactive-PCP) prover depends only on the preceding $O(\log n)$ bits sent by the verifier, implies that the interactive-PCP system can be converted into a *history-ignorant* one, where the length of the longest message sent by the verifier in the interactive phase is $\lambda = O(\log n)$. This increases the communication complexity ℓ by at most a quadratic factor, and does not change the other parameters. The fact that the interactive phase of the (original) interactive-PCP is public-coin, implies that in the resulting (history-ignorant) interactive-PCP, each message sent by the verifier depends only on the verifier's random coin tosses, and can be computed in time $\leq \text{poly}(\ell)$.

Applying Theorem 1 (with security parameter $t' = \text{poly}(\ell, t)$) to this interactive-PCP system, results with a PCA system for L, with the desired parameters, where the prover runs in time $\text{poly}(n)$.

References

1. Aiello, W., Bhatt, S.N., Ostrovsky, R., Rajagopalan, S.: Fast Verification of Any Remote Procedure Call: Short Witness-Indistinguishable One-Round Proofs for NP. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, p. 463. Springer, Heidelberg (2000) (manuscript withdrown by the authors prior to ICALP)
2. Arora, S., Lund, C., Motwani, R., Sudan. M., Szegedy, M.: Proof Verification and Hardness of Approximation Problems. In: FOCS 1992, pp. 14–23 (1992); also in J. ACM 45(3), 501–555 (1998)
3. Arora, S., Safra, S.: Probabilistic Checking of Proofs: A New Characterization of NP. In: FOCS 1992, pp. 2–13 (1992); also in J. ACM 45(1), 70–122 (1998)
4. Barak, B.: How to Go Beyond the Black-Box Simulation Barrier. In: FOCS 2001, pp. 106–115 (2001)
5. Babai, L., Fortnow, L., Lund, C.: Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. In: FOCS 1990, pp. 16–25 (1990); also In Computational Complexity 1, 3–40 (1991)
6. Cachin, C., Micali, S., Stadler, M.: Computationally Private Information Retrieval with Polylogarithmic Communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
7. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private Information Retrieval. J. ACM 45(6), 965–981 (1998)
8. Damgard, I., Fazio, N., Nicolosi, A.: Non-interactive Zero-Knowledge from Homomorphic Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 41–59. Springer, Heidelberg (2006)
9. Dwork, C., Langberg, M., Naor, M., Nissim, K., Reingold, O.: Succinct Proofs for NP and Spooky Interactions (unpublished manuscript)
10. Feige, U., Goldwasser, S., Lovasz, L., Safra, S., Szegedy, M.: Interactive Proofs and the Hardness of Approximating Cliques. In: FOCS 1991, pp. 2–12 (1991); also in J. ACM 43(2), 268–292 (1996)
11. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
12. Fortnow, L., Santhanam, R.: Infeasibility of Instance Compression and Succinct PCPs for NP. In: STOC 2008, pp. 133–142 (2008)

13. Goldwasser, S., Kalai, Y.T.: On the (In)security of the Fiat-Shamir Paradigm. In: FOCS 2003 (2003)
14. Goldwasser, S., Kalai, Y.T., Rothblum, G.: Delegating Computation: Interactive Proofs for Muggles. In: FOCS 2007, pp. 113–122 (2007)
15. Harnik, H., Naor, M.: On the Compressibility of NP instances and Cryptographic Applications. In: FOCS 2006, pp. 719–728 (2006)
16. Kushilevitz, E., Ostrovsky, R.: Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In: FOCS 1997, pp. 364–373 (1997)
17. Kalai, Y.T., Raz, R.: Succinct Non-Interactive Zero-Knowledge Proofs with Preprocessing for LOGSNP. In: FOCS 2006, pp. 355–366 (2006)
18. Kalai, Y.T., Raz, R.: Interactive PCP. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 536–547. Springer, Heidelberg (2008)
19. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC 1992, pp. 723–732 (1992)
20. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic Methods for Interactive Proof Systems. In: FOCS 1990, pp. 2–10 (1990); also in J. ACM 39(4), 859–868 (1992)
21. Micali, S.: CS Proofs (Extended Abstracts). In: FOCS 1994, pp. 436–453 (1994)
22. Shamir, A.: IP=PSPACE. J. ACM 39(4), 869–877 (1992)

On the Composition of Public-Coin Zero-Knowledge Protocols

Rafael Pass[1], Wei-Lung Dustin Tseng[1], and Douglas Wikström[2]

[1] Cornell University, USA
[2] KTH Royal Institute of Technology, Sweden

Abstract. We show that only languages in BPP have public-coin, black-box zero-knowledge protocols that are secure under an unbounded (polynomial) number of parallel repetitions. This result holds both in the plain model (without any set-up) and in the Bare Public-Key Model (where the prover and the verifier have registered public keys). We complement this result by showing the existence of a public-coin black-box zero-knowledge proof that remains secure under any *a-priori* bounded number of concurrent executions.

1 Introduction

Zero-knowledge (ZK) interactive protocols [GMR89] are paradoxical constructs that allow one player P (called the prover) to convince another player V (called the verifier) of the validity of a mathematical statement $x \in L$, while providing *zero additional knowledge* to the verifier. Beyond being fascinating in their own right, ZK proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks. A fundamental question regarding zero-knowledge protocols is whether their composition remains zero-knowledge. The three most basic notions of compositions are sequential composition [GMR89, GO94], parallel composition [FS90, GK96b] and concurrent composition [FS90, DNS04]. In a sequential composition, the players sequentially run many instances of a zero-knowledge protocol, one after the other. In a parallel composition, the instances instead proceed in parallel, at the same pace. Finally, in a concurrent composition, messages from different instances of the protocol may be arbitrarily interleaved.

While the definition of ZK is closed under sequential composition [GO94], this no longer holds for parallel composition [GK96b] (and thus not for concurrent composition either). However, there are zero-knowledge protocols for all of NP that have been demonstrated to be secure under both parallel and concurrent composition. For the case of only parallel composition, constant-round protocols are known [Gol02, FS90, GK96a]. For the case of concurrent composition, a series of work [RK99, KP01, PRS02] show feasibility of $\tilde{O}(\log n)$-round protocols; furthermore, this round-complexity is essentially optimal with respect to black-box simulation [KPR98, Ros00, CKPR01].

Whereas the original ZK protocols of [GMR89, GMW91, Blu87] are *public-coin*—i.e., the verifier's messages are its random coin-tosses—all of the aforementioned parallel or concurrent ZK protocols use private coins. Indeed, in their seminal paper, Goldreich and Krawczyk [GK96b] (GK from now on) showed that only languages in BPP have *constant-round* public-coin (stand-alone) black-box ZK protocols with negligible soundness error, let alone the question of parallel composition. In particular, their results imply that (unless NP $\subseteq$ BPP) the constant-round ZK protocols of e.g., [GMW91, Blu87] with constant soundness error cannot be black-box ZK under parallel repetition (as this would yield a constant-round black-box ZK protocol with negligible soundness error).

A natural question is whether the constant-round restriction imposed by the GK result is necessary. Namely,

Is there a (possibly super-constant round) public-coin black-box ZK protocol that is secure under parallel (or even concurrent) composition?

Our results. In this work, we provide a negative answer to the above question. Namely, we show that only languages in BPP have public-coin black-box ZK protocols that remain secure under parallel (and thus also concurrent) composition. Thus, whereas for private-coin protocols, a super constant number of rounds helps in establishing concurrent composition [RK99, KP01, PRS02], we conclude that it is not the case for public-coin protocols.

Theorem (Informal). *If L has a public-coin argument that is black-box parallel ZK, then $L \in$ BPP.*

In fact, our result establishes that any black-box ZK protocol that remains secure under m parallel executions must have $\tilde{\Omega}(m^{1/2})$ rounds.

On the positive side we show that every language in NP has a public-coin black-box ZK proof that remains secure under an *a-priori* bounded number of concurrent (and thus parallel) executions.

Theorem (Informal). *Assume the existence of one-way functions. Then for every polynomial m, there exists an $O(m^3)$-round public-coin black-box m-bounded concurrent ZK proof for NP.*

This complements a result of [Bar01], which constructs constant-round public-coin bounded-concurrent ZK *arguments* (rather than proofs) using *non-black-box* simulation.

We next turn to compositions in models with trusted set-up. Canetti, Goldreich, Goldwasser and Micali [CGGM00] show that in the Bare Public-Key (BPK) Model, where each player has a registered public-key, constant-round black-box concurrent ZK protocols exist for all of NP (whereas in the plain model without set-up, as mentioned above, $\tilde{\Omega}(\log n)$ rounds are necessary for non-trivial languages [CKPR01]). We show that for the case of public-coin protocols, the BPK set-up does not help with composition.

Theorem (Informal). *If L has a public-coin argument in the BPK model that is black-box parallel ZK, then $L \in$ BPP.*

Finally, as we will see, some of the intermediate ideas in our work are closely related to the notion of *resettable soundness* [BGGL01]. Very informally, we establish that parallel repetition of public-coin protocols not only reduces the soundness error [PV07, HPPW08], but also *qualitatively* strengthen the soundness—the new protocols will be secure even under a "resetting" attack.

Techniques. To describe our technique, let us first briefly recall the GK lower bound showing that only languages in L have $O(1)$-round public-coin black-box ZK. Let (P, V) be a black-box ZK proof of a language L. Consider a malicious verifier V^* which, instead of picking its messages at random, computes them by applying a hash function to the current transcript. GK show that any black-box simulator S, together with V^* can decide L: on input x, simply run $S^{V^*(x)}(x)$ and accept if S outputs a view where V^* is accepting. It easily follows that if $x \in L$, then $S^{V^*}(x)$ will output a transcript where V^* is accepting (as the honest prover would convince V^*). The crux of their proof is to show that if $x \notin L$, then $S^{V^*}(x)$ will output an accepting view only with small probability. If S was not rewinding V^* this would directly follow from the soundness pf (P, V). However S might rewind V^*, and might only convince V^* in one of its rewinding "threads". Nonetheless, GK manages to show that if S—using rewinding, or "resetting"—manages to convince V^*, then we can construct a machine T that uses S to convince an external verifier V (without rewinding), contradicting the soundness of (P, V). In other words, they show that the protocol (P, V^*) is sound under a "resetting-attack" [CGGM00, BGGL01]. Analogously, to prove our results, we show that if we have take a public-coin interactive proof (P, V) and repeat it sufficiently many times in parallel (and again letting the verifier pick its messages by applying a hash function to the transcript), then the resulting protocol is sound under a resetting-attack.

More details on the reduction. GK, as well as all subsequent black-box lower bounds e.g., ([KPR98, Ros00, CKPR01, BL02, Kat08, HRS09]) rely on the following approach for constructing the stand-alone (non-resetting) prover T (given the "rewinding" simulator S). T incorporates S and internally emulates the execution of S with an internally emulated verifier (which of course can be rewound). While doing this emulation, T also appropriately picks some messages sent by S to the internal verifier, and forwards them externally (and also forwards back the reply received externally). The crux of the various lower bounds is how these messages (to be forwarded externally) are chosen. The difficulty of this task stems from the fact that, at the time of deciding whether to externally forward a message or not, T does not yet know if the simulator will eventually choose this message to "continue" its simulation, or treat this messages simply as a "rewinding" (used to collect information).

For the case of constant-round protocols, GK show that a *random* selection of messages to forward externally works. (If the protocol has d rounds, this random selection is "correct" with probability $1/m^d$, where m is the number of queries made by the simulator to its verifier.) To handle a super constant number of rounds, Canetti, Kilian, Petrank and Rosen [CKPR01] show that when dealing with an adversarial verifier that can schedule messages in an arbitrary way, there exists some particular scheduling which makes it easy to identify appropriate messages to forward externally (as long as the number of rounds is sub-logarthmic).

This general approach of simply running the simulator S "straight-line" seems hard to extend to protocols with a polynomial number of rounds; the number of possible choices for messages to forward to the external verifier becomes too large. To get around this problem, we use a different technique. Instead of simply running S "straight-line", we let T *rewind* S (while S itself believes it is rewinding the verifier). This makes it possible for us to "check" whether a message is "good" before forwarding it to the external verifier. Our strategy is twofold. First, we only externally forward queries that have a good chance of being included in the output view; since the protocol is public-coin, we can estimate this chance by doing "test-runs" of S. Once we have forwarded a query, we will force S to include it in the output view by repeatedly rewinding S. Intuitively, if the forwarded queries are indeed "good", then T should break the soundness of Π after polynomially many rewinds. But what if the external verifier acts differently from our test-runs and replies with a "bad" response to a forwarded query? Using a probabilistic lemma due to Ran Raz [Raz98] (used to prove that parallel repetition reduces the soundness error in two-prover games) we can show that if we have enough parallel sessions, and the external verifier only decides the verifier response in one session, the "goodness" of a forwarded query will not change much.

We remark that our approach shares similarities with previous works on the topic of soundness amplification under parallel repetitions, such as [BIN97] [PV07] [IJK07], and especially [HPPW08]; in particular, our use of Raz's lemma is similar to its use in [HPPW08]. However, whereas those works show how to transform a parallel prover with "small" success probability into a stand-alone prover with "high" success probability, we show how to transform a *rewinding* parallel prover into a (non-rewinding) stand-alone prover. This requires overcoming several novel obstacles: most notably, we are required to deal with the difficulty of forcing the simulator S to output a view which uses the queries externally forwarded by T.

To extend our lower bound to the BPK model, we run into the additional problem that the external verifier can decide whether to accept or reject based on its secret key (which T does not know). T can thus no longer determine whether an external verifier accepts or rejects when doing test-runs, which is crucial for deciding which messages to forward externally. By relying on the "trust-halving" technique from [IW97, BIN97], and its refinement in [HPPW08], we show how T can make an "educated" guess which is sufficiently good.

Parallel-repetition and Resettable-soundness. As an independent contribution, we believe that our techniques elucidates an intriguing (and useful) connection between *lower bounds* for black-box ZK, and feasibility results for soundness/hardness amplification. As mentioned, our core technical contribution shows that (appropriate) parallel-repetition of a public-coin protocol not only reduced the soundness error [PV07, HPPW08], but also yields a protocol that is sound even under a resetting attack [GK96b, CGGM00, BGGL01]. To establish our ZK lowerbound, we only consider a "weak" notion of "resettable-soundness" where the statement to be proved cannot be changed. In the full version of the paper, we shows that if the original protocol also is a *proof of* *knowledge* [GMR89, FS90, BG02], then the parallelized version also satisfies the stronger notion of resettable-soundness from [BGGL01] (where the adversary can also change the statement during its rewindings). [BGGL01] showed a similar type of result for $O(1)$-round public-coin proofs of knowledge.

Outline. We introduce some preliminaries in Sect. 2, and jump into our impossibility results in Sect. 3. Next we present our public-coin bounded-concurrent zero-knowledge protocol in Sect. 4. Details of our extension to the Bare Public Key model and our application to resettable soundness can be found in the full version of this paper.

2 Preliminaries

We assume familiarity with indistinguishability, interactive proofs and commitments. $|x|$ denotes the length of a (bit) string x, and $[n]$ denotes the set $\{1, \dots, n\}$.

Let $\Pi = \langle P, V \rangle$ be an interactive proof for a language L between prover P and verifier V. We assume WLOG that Π starts with a verifier message and ends with a prover message, and say Π has k **rounds** if the prover and verifier each sends k messages alternatively. The notation $\langle v_1, p_1, \dots \rangle$ specifies a full or partial **transcript** of Π where v denotes verifier messages and p denotes prover messages. Π is **public-coin** if the verifier messages are just independent segments of V's random tape.

We may repeat an interactive proof in parallel. Let $\Pi^m = \langle P^m, V^m \rangle$ be Π repeated in m **parallel sessions**; that is, each prover and verifier message in Π^m is just concatenation of m copies of the corresponding message in Π. V^m completes Π in all m sessions (or abort in all sessions), and accepts if and only if all m sessions are accepted by V.

In general, an adversarial verifier is not restricted to parallel schedules. An m-session **concurrent adversarial verifier** V^* is a probabilistic polynomial time machine that, on common input x and auxiliary input z, interacts with $m(|x|)$ independent copies of P concurrently (called **sessions**). There are no restrictions on how V^* schedules the messages among the different sessions, and V^* may choose to abort some sessions but not others. Let $\mathrm{View}^P_{V^*}(x, z)$ be the random variable that denotes the **view** of V^* in an interaction with P (this includes the random coins of V^* and the messages received by V^*). *Note that for public-coin protocols, the view of V^* is just the transcript of the interaction.*

A **black-box simulator** S is a probabilistic polynomial time machine that is given black-box access to V^* (written as $S = S^{V^*}$). Formally, the random tape of V^*, denoted by r, is uniformly chosen and fixed a priori, and S is allowed to specify a valid partial transcript $\tau = \langle v_1, p_1, \dots, p_i \rangle$ of $\langle P, V^*_r \rangle$, and query V^*_r for the next verifier message v_{i+1}. Here, τ is **valid** if it is consistent with V^*_r — i.e., each verifier message v_j in τ is what V^* would have responded given the previous prover messages $p_1, \dots, p_{j-1}$ (and the fixed random tape r). Note that S is allowed to "rewind" V^* by querying V^* with different partial transcripts that shares a common prefix.

Intuitively, an interactive proof is zero-knowledge (ZK) if the view of any (stand-alone) adversarial verifier V^* can be generated by a simulator. The protocol is concurrent ZK if the view of any concurrent adversarial verifier can be generated as well. The formal definitions follow.

Definition 1 (Black-Box Zero-Knowledge [GMR89, GO94]). *Let $\Pi = \langle P, V \rangle$ be an interactive proof (or argument) for a language L. Π is black-box zero-knowledge if there exists a black-box simulator S such that for every common input x, auxiliary input z, random tape r, and every (stand-alone) adversary V^*, $S^{V_r^*(x,z)}(x)$ runs in time polynomial in $|x|$. Furthermore, the ensembles $\{\mathrm{View}_{V_r^*}^{P}(x,z)\}_{x\in L,z,r\in\{0,1\}^*}$ and $\{S^{V_r^*(x,z)}(x)\}_{x\in L,z,r\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

Definition 2 (Black-Box Concurrent Zero-Knowledge [DNS04]). *Let $\Pi = \langle P, V \rangle$ be an interactive proof (or argument) for a language L. Π is black-box concurrent zero-knowledge if for all polynomials m, there exists a black-box simulator S_m such that for every common input x, auxiliary input z, random tape r and every m-session concurrent adversary V^*, $S_m^{V_r^*(x,z)}(x)$ runs in time polynomial in $|x|$. Furthermore, the ensembles $\{\mathrm{View}_{V_r^*}^{P}(x,z)\}_{x\in L,z,r\in\{0,1\}^*}$ and $\{S_m^{V_r^*(x,z)}(x)\}_{x\in L,z,r\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

We also consider a bounded version of concurrent zero-knowledge where the order of quantifiers are reversed [Bar01].

Definition 3 (Black-Box Bounded Concurrent Zero-Knowledge). *Let $\Pi = \langle P, V \rangle$ be an interactive proof (or argument) for a language L and let m be a polynomial. Π is black-box m-bounded concurrent zero-knowledge if there exists a black-box simulator S such that for every common input x, auxiliary input z, random tape r, and every m-session concurrent adversary V^*, $S^{V_r^*(x,z)}(x)$ runs in time polynomial in $|x|$. Furthermore, the ensembles $\{\mathrm{View}_{V_r^*}^{P}(x,z)\}_{x\in L,z,r\in\{0,1\}^*}$ and $\{S^{V_r^*(x,z)}(x)\}_{x\in L,z,r\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

3 Impossibility

From now on zero-knowledge refers to *black-box* zero-knowledge. In this section we show that only languages in BPP have public-coin concurrent zero-knowledge protocols. We actually show a stronger result: Except for languages in BPP, no public-coin protocols remains black-box zero-knowledge when repeated in parallel. The formal theorems are stated below, where n denote the security parameter or the input size.

Theorem 1. *Suppose language L has a $k = \mathrm{poly}(n)$-round public coin black-box zero-knowledge proof Π with soundness error $1/2$. If $m \geq k \log^2 n$ and Π^m is zero-knowledge, then $L \in$ BPP.*

Theorem 2. *Suppose language L has a $k = \mathrm{poly}(n)$-round public-coin black-box zero-knowledge argument Π with soundness error $1/2$. If $m \geq (k^2 \log k) \log^2 n$ and Π^m is zero-knowledge, then $L \in$ BPP.*

We remark here that our theorems also hold with respect to so-called non-aborting verifiers that never send an invalid message.

3.1 Common Proof Components

The proofs of Theorem 1 and 2 begin in the same high-level framework as that of [GK96b]. Suppose a language L has a public-coin ZK protocol $\Pi = \langle P, V \rangle$, and Π^m is zero-knowledge with a black-box simulator S that runs in time n^d. To show that $L \in \mathsf{BPP}$, we construct a "random-looking" adversarial verifier, V^*, and consider the following decision algorithm D: $D(x)$ runs S^{V^*} to generate a view of V^*, and accepts x if and only if V^* accepts given the generated view (which in turn occurs if and only if the honest verifier V accepts in all m sessions of the view).

V^* is actually a family of adversarial verifiers constructed as follows. Let H be a family of hash functions that is random enough compared to the running time of S; formally, H should be n^d-wise independent (see [GK96b, CG89]). Let V_h^* be the verifier that when queried with transcript τ, responds (deterministically) with the message $h(\tau)$. We write V^* to mean V_h^* for a randomly chosen h, i.e., when D runs S^{V^*}, D first chooses h randomly from H and then run $S^{V_h^*}$.

We make two easy observations about S^{V^*}. First, we may assume that whenever S queries V^* with a transcript or outputs a transcript τ, it first queries V^* with all the prefixes of τ; this only increases the running time of S polynomially. Second, we may assume that S never queries V^* with the same transcript twice (instead S may keep a table of answers). Then the set of all responses generated by V^* is identical to the uniform distribution since H is n^d-independent and S makes at most n^d queries to V^*.

We need to show that decision procedure D is both complete and sound. Completeness states that if $x \in L$, then D should accept x with probability at least 2/3. This easily follows: The output of $S^{V^*}(x)$ is indistinguishable from the interaction of $\langle P^m, V^* \rangle$ since S is a zero-knowledge simulator. Furthermore, $\langle P^m, V^* \rangle$ is identical to m-copies of $\langle P, V \rangle$ since V^* produces independent, truly random verifier messages. Finally, by the completeness property of Π, V will accept x with probability 1 in all the copies of $\langle P, V \rangle$.

Soundness states that if $x \notin L$, then D should accept with probability at most 1/3. That is, $S^{V^*}(x)$ can produce an accepting view of V^* with probability at most 1/3. In a sense, this is the soundness of protocol Π^m against a rewinding prover such as S. This property is shown separately for proofs and arguments in the next two sections.

3.2 Proof of Theorem 1: Zero-Knowledge Proofs

We show that D is sound when Π is a proof. Our approach follows that of [GK96b] while relying on the soundness amplification theorem of [BM88].

Suppose for the sake of contradiction that for some $x \notin L$, $S^{V^*}(x)$ produces an accepting view with probability more than 1/3; we will use S to lower bound the soundness error of Π^m (as an interactive proof of L). Whenever S^{V^*} outputs a valid accepting view of V^* and a corresponding transcript τ of Π^m, S would have queried V^* for each of the k verifier messages in τ (recall that we assumed this without loss of generality). A cheating prover of Π^m can therefore run S^{V^*} internally, guess which queries of S are used to form the accepting output transcript, and forward them to an outside honest verifier of Π^m. Since S has maximum running time n^d, it can only query V^* for at most n^d messages. The probability of guessing all the right queries is then at least n^{-dk} (one guess for each round of Π). Note that forwarding queries to an outside honest verifier does not lower the acceptance probability of S^{V^*} since V^* is identical to a honest verifier (they both respond with random messages). Thus this cheating prover, using S, can break the soundness of Π^m with probability at least $(1/3)n^{-dk} = \Omega(2^{-dk \log n})$.

On the other hand, recall that Π has soundness error less than 1/2. By the soundness amplification theorem of [BM88], Π^m should have soundness error at most $O(2^{-m})$. Since $m \geq k \log^2 n$, we have $O(2^{-m}) < \Omega(2^{-dk \log n})$ and reach a contradiction. □

3.3 Proof of Theorem 2: Zero-Knowledge Arguments

We now show that D is sound even when Π is an argument. Again we argue by contradiction, and suppose $S^{V^*}(x)$ outputs an accepting view for some $x \notin L$ with probably more than 1/3. We cannot repeat the proof of Theorem 1 because parallel repetitions cannot reduce the soundness of arguments beyond being negligibly small. Therefore we cannot use S to break the soundness of Π^m. Instead, we directly show a parallel repetition theorem for "resettable-soundness"; that is, we relate the "resettable" soundness of $\langle P^m, V^* \rangle$ to the soundness of Π.

Proof Outline. The rest of this section describes how to construct a cheating prover T for Π. T runs S internally and plays the role of V^* when S makes a query. To break the soundness of Π, T needs to choose one of the m sessions and forward a complete set of S queries in that session (one for each round of Π) to an honest outside verifier V of Π. Moreover, S must eventually use these forwarded queries to output an accepting view. This is challenging since S may query V^* multiple times for each round of Π^m. While T must decide to forward a query or not at the time of the query, S can wait until all queries are completed before choosing which queries to form the output view. To overcome this obstacle, a key part of our analysis relies on *rewinding* S (note that at the same time, S believes that it is rewinding V^*). Our strategy is twofold. First we only forward queries that has some chance (preferably a good chance) of being included in the output view; this is done by doing test-runs of S. Once we have forwarded a query, we will force S to include it in the output view by repeatedly rewinding S.

We can describe a **transcript** of S as an alternating sequence of queries from S and responses from T, $[s_1, t_1, s_2, t_2, \ldots]$, where each S-query is in fact a partial transcript of Π^m that ends with a prover message (awaiting a verifier response). To avoid confusion, in our analysis τ and $\langle \cdot \rangle$ denote views of V^* (which are just transcripts of Π^m), while h and $[\cdot]$ denote transcripts of S. Recall that S may rewind V^*, and thus a transcript of S may be much longer than a view of V^*. Since the randomness of S is fixed, the behaviour of S is entirely determined by the T-responses in a transcript. The goal of T is then to generate a full transcript of S so that S produces an accepting view of V^* and a corresponding transcript

τ of Π^m, while simultaneously having the foresight to forward (a session of) all the S-queries pertaining to τ to the external verifier V (i.e. all S-queries that are a prefix of τ). If so, T has broken the soundness of Π, and we call this a **successful** simulation of S.

On a high level, T first fixes a random session $j_0 \in 1, \ldots, m$ as the forwarding session. Then, T will incrementally fix the transcript of S while forwarding S-queries to V in k iterations (one for each round of Π). During each iteration, T first forwards a query to V. Then, T continues the simulation of S using V's response until it finds a suitable query to forward in the next iteration. In more details:

Step 1. In iteration i, T starts with a partial transcript h_i of S that ends with a query for the i^{th} message of Π. T will forward session j_0 of this query to V and receive a reply $v_i^{j_0}$.

Step 2. Fixing the reply $v_i^{j_0}$, T randomly completes the partial transcript h_i up to $300k^2n^d$ times until it finds a successful completion h. If no successful completion is found, T aborts. Otherwise, let τ be the accepting view of V^* produced by S under transcript h (in particular, τ must include all the queries that have been forwarded by T). By assumption, S must query T for the $i + 1^{\text{st}}$ verifier message in τ. Let h_{i+1} be the prefix of h up to this query (note that h_{i+1} is an extension of h_i), and this query (considered a "good" query for the $i + 1^{\text{st}}$ verifier message) will be forwarded in the next iteration.

During the analysis, we first use Raz's lemma to show that because the number of sessions is large and j_0 was chosen randomly, we may pretend $v_i^{j_0}$ is nicely chosen conditioned on success, just like the other sessions (chosen by T in step 2). We also show that T rarely aborts.

Proof Details. We now introduce a series of hybrid simulators that formally defines and analyses T; all our hybrids will always generate truly random responses to S-queries so that S cannot distinguish the hybrids from V^*. We will start with a hypothetical hybrid, and gradually move towards T.

Hybrid 1. Our first hybrid $T^{(1)}$ serves to introduce the general idea of how T queries S internally; $T^{(1)}$ does not yet forward messages to the external verifier V.

$T^{(1)}$ builds a full transcript of S in $k + 1$ iterations. In iteration i, $T^{(1)}$ fixes an S-query τ_i for the i^{th} message of Π^m. This query should have a good chance of being included by S in an accepting transcript of Π^m, and therefore is a good candidate to forward externally. Note that fixing an S-query amounts to fixing the transcript of S up until the desired S-query is made.

We now describe $T^{(1)}$ in detail. In the very beginning, $T^{(1)}$ fixes a random session $j_0 \in \{1, \ldots, m\}$; eventually the j_0^{th} session will be forwarded externally. After that, $T^{(1)}$ incrementally grows a transcript of S in k iterations. During the i^{th} iteration, $T^{(1)}$ receives a partial transcript of S from the previous iteration, $h_i = [t_1, s_1, \ldots, s_\ell = \tau_i]$, where τ_i is a S-query for the i^{th} verifier message of Π^m ($h_1 = []$, the empty transcript). Looking ahead, session j_0 of τ_i will be forwarded to the external V in later hybrids of T. As an invariant maintained by $T^{(1)}$, it should be possible to extend h_i into a full transcript of S where S outputs an accepting view of V^* containing the query τ_i. We call such a full transcript a **successful** completion of h_i. Each iteration can be further divided into two steps:

Step 1. $T^{(1)}$ does not forward τ_i to the external V; instead it simulates a response to its liking. $T^{(1)}$ randomly samples a completion of h_i into h conditioned on success (always possible due to the invariant). Let $v_i^{(j_0)}$ be the response to τ_i in the j_0^{th} session in the successful completion h; it is used in place of a response from V. Let $\tilde{h}_i$ be a slight extension of the partial transcript h_i where the session j_0 response to τ_i is fixed to $v_i^{(j_0)}$.

Step 2. $T^{(1)}$ now samples a completion of $\tilde{h}_i$ into $\tilde{h}$ conditioned on success (note that h from the previous step is one such completion). Under $\tilde{h}$, S would output an accepting view τ of V^* (note that τ must extend τ_i). Let τ_{i+1} be the S query for the $i+1^{\text{st}}$ verifier message in τ (note that τ_{i+1} extends τ_i). $T^{(1)}$ then sets h_{i+1} to be the prefix of $\tilde{h}$ up to when S makes the query τ_i. Note that the invariant holds since by definition $\tilde{h}$ is a successful completion of h_{i+1}.

Note that in Step 2 of the final (k^{th}) iteration, $T^{(1)}$ simply outputs $\tilde{h}$ as a full transcript of S (there is no τ_{k+1} to fix). Due to the invariant, $T^{(1)}$ always produce a transcript of S where S outputs an accepting transcript τ, whose incremental prefixes $\tau_1, \dots, \tau_k$ were forwarded by T to the external verifier.

Hybrid 2. Our second hybrid, $T^{(2)}$, describes a way to sample successful completions in Step 2 of each iteration (Step 1 will be replaced with the external verifier and is left alone for now). In Step 2, $T^{(2)}$ randomly completes the given partial execution ($\tilde{h}_i$) up to $300k^2n^d$ times, until a successful completion is found. If none of the completions are successful, $T^{(2)}$ aborts. Note that conditioned on $T^{(2)}$ not aborting, the output distribution of $T^{(2)}$ is *identical* to $T^{(1)}$.

To show that $T^{(2)}$ aborts with small probability, suppose for now that $T^{(2)}$ is allowed to sample an unbounded number of completions. Let us bound the expected number of completions that are needed to have a successful one. In the following analysis we distinguish between two probability spaces: $\Pr_S[\cdot]$ is used to measure probabilities over a single execution of S. On the other hand, $\Pr_T[\cdot]$ is used to measure probabilities over an execution of $T^{(2)}$ (with unbounded number of completions) which includes rewinding and executing S multiple times.

Let H_i and $\tilde{H}_i$ be the set of possible partial transcripts of S that is given to $T^{(2)}$ in Step 1 and Step 2 of the i^{th} iteration, respectively. Given $h \in H_i$ (or $\tilde{H}_i$), let $\Pr_S[h]$ denote the probability that a transcript of S has prefix h, and let $\Pr_T[h]$ denote the probability that $T^{(2)}$ is given h in the i^{th} iteration; similarly, $\Pr_S[\cdot \mid h]$ and $\Pr_T[\cdot \mid h]$ are probabilities conditioned on these events occurring. Let A^h be the event (over the S probability space) that a transcript of S has prefix h and is a successful completion of h; as a special case, $A = A^{\emptyset}$ is just the event that S outputs an accepting transcript. Also let R_i be the random variable (over the $T^{(2)}$ probability space) that denotes the number of completions performed by $T^{(2)}$ in step 2 of iteration i.

Lemma 3. $\mathbb{E}_T[R_i] \leq 3n^d$.

Proof. First expand $\mathbb{E}_T[R_i]$ by conditioning on the transcript h fixed in Step 1:

$$\mathbb{E}_T[R_i] = \sum_{h \in \tilde{H}_i} \Pr_T[h]\mathbb{E}_T[R_i \mid h] \tag{1}$$

Recall that in Step 2, $T^{(2)}$ samples random completions of h until a successful completion is found. Therefore

$$\mathbb{E}_T[R_i \mid h] = \frac{1}{\Pr_S[A^h \mid h]} \;\Rightarrow\; \mathbb{E}_T[R_i] = \sum_{h \in \tilde{H}_i} \Pr_T[h]\frac{1}{\Pr_S[A^h \mid h]} \tag{2}$$

We now state a claim and give its proof in the full version of this paper.

Claim 4. *Let $h \in \tilde{H}_i$.* $\Pr_T[h]\Pr_S[A] = \Pr_S[A^h]$.

Intuitively, the claim says that the probability of $T^{(2)}$ fixing h is proportional to the probability of successfully completing h ($\Pr_S[A]$, the probability that S produces an accepting transcript, is the normalizing factor). This can be shown by a counting argument. By expanding the RHS of Claim 4 and rearranging terms, we have

$$\begin{aligned} \Pr_T[h]\Pr_S[A] = \Pr_S[A^h] &= \Pr_S[h]\Pr_S[A^h \mid h] \\ \Rightarrow\; \Pr_T[h]\frac{1}{\Pr_S[A^h \mid h]} &= \Pr_S[h]\frac{1}{\Pr_S[A]} \leq 3\Pr_S[h] \end{aligned}$$

since we assumed $\Pr_S[A] \geq 1/3$. Substituting this back into (2) gives

$$\mathbb{E}_T[R_i] \leq 3\sum_{h \in \tilde{H}_i} \Pr_S[h] \tag{3}$$

finally, we may breakup the set $\tilde{H}_i$ based on the length of h which ranges from 1 to n^d (where length is the number of S-queries). Since each transcript of S has exactly one length ℓ prefix:

$$\mathbb{E}_T[R_i] \;\leq\; 3\sum_{\ell=1}^{n^d} \sum_{h \in \tilde{H}_i, |h|=\ell} \Pr_S[h] \;\leq\; 3\sum_{\ell=1}^{n^d} 1 = 3n^d$$

□

Now we can show that $300k^2n^d$ random completions are enough for $T^{(2)}$.

Lemma 5. $T^{(2)}$ *aborts with probability at most* $1/5$.

Proof. Since $\mathbb{E}_T[R_i] = \sum_{\tilde{h}_i} \Pr_T[\tilde{h}_i]\mathbb{E}_T[R_i \mid \tilde{h}_i] = \mathbb{E}_T[\mathbb{E}_T[R_i \mid \tilde{h}_i]] \leq 3n^d$, the Markov inequality states that the probability of $T^{(2)}$ fixing an $\tilde{h}_i$ such that $\mathbb{E}_T[R_i \mid \tilde{h}_i] \geq 30kn^d$ is at most $1/(10k)$. For each "good" $\tilde{h}_i$ where $\mathbb{E}_T[R_i \mid \tilde{h}_i] < 30kn^d$, we apply the Markov inequality again to obtain $\Pr_T[R_i \geq 300k^2n^d \mid h_i] \leq 1/(10k)$. Using the union bound we see that in any iteration, $T^{(2)}$ aborts in Step 1 with probability at most $1/(5k)$. A final union bound over k iterations of Step 2 shows that $T^{(2)}$ aborts overall with probability at most $1/5$. □

Hybrid 3. Our third and final hybrid $T^{(3)} = T$ differs from $T^{(2)}$ in Step 1 of each iteration. Recall that some session j_0 is chosen randomly as the forwarding session. Instead of generating $v_i^{(j_0)}$ in Step 1, $T^{(3)}$ asks the external honest verifier V for a verifier message. Because Π is public-coin, $T^{(3)}$ can continue to complete partial transcripts of S even if session j_0 is forwarded to V externally.

Given transcript $h_i = [t_1, s_1, \ldots, s_\ell = \tau_i]$ in iteration i, $T^{(3)}$ forwards session j_0 of τ_i to V, and uses the response from V as $v_i^{(j_0)}$ in Step 2. Suppose for now that $T^{(3)}$ does not abort and terminates successfully. Then S would have generated an accepting transcript τ of Π^m. Since $\tau_1, \ldots, \tau_k$ are prefixes of τ, session j_0 of τ would be an accepting transcript of Π consisting of forwarded prover messages and responses from V. This breaks the soundness of Π.

Therefore, it remains to show that $T^{(3)}$ is successful with probability more than 1/2. We will use Raz's lemma [Raz98] in analogy with [IJK07, HPPW08] to show that $v_i^{(j_0)}$ as generated by $T^{(1)}$ and $T^{(2)}$ is actually very close to the uniformly random messages generated by the honest verifier V. First we cite Raz's lemma as it appears in [Hol07, Lemma 5]:

Lemma 6. *Let $\{U_j\}_{j\in[m]}$ be independent random variables on $\mathcal{U}$ with probability distribution P_{U_j}. Let W be an event in $\mathcal{U}^m$ and $\Pr[W]$ be measured according to the joint probability distribution $\Pi_j P_{U_j}$. Then*

$$\sum_{j=1}^{m} \Delta(U_j|W, U_j) \leq \sqrt{m \log\left(\frac{1}{\Pr[W]}\right)}$$

where Δ is the statistical distance between distributions, and $U_j|W$ is the j^{th} component of an element in $\mathcal{U}^m$ chosen based on the joint probability distribution $\Pi_j P_{U_j}$, conditioned on W.

In other words, let $\{U_j\}_j$ be independent random variables, and let W be an event over $\Pi_j U_j$. If W occurs with high probability and there are many U_j, then on average over j, sampling U_j conditioned on W does not differ much from simply sampling U_j. Lemma 6 allows us the bound the change in success probability when $T^{(3)}$ forwards messages from a random session to V.

Lemma 7. *$T^{(3)}$ fails with probability at most $3/10 + O(1/\log n)$.*

Proof. We first construct a series of finer hybrids, $T_1, \ldots, T_{k+1}$, where T_i proceeds as $T^{(2)}$ until the start of iteration i (no forwarding), and continues as $T^{(3)}$ afterwards (with forwarding)[1]. Observe that $T_1 = T^{(3)}$ and $T_{k+1} = T^{(2)}$.

Consider two neighboring hybrids, T_i and T_{i+1}, which differ only in iteration i. Let h be the partial execution given in iteration i. For $j \in [m]$, let U_j be the random variable that denotes all the additional session j messages sent by T to randomly complete h, i.e., $\{U_j\}_j$ are independent and uniformly random. Let W^h be the event that the random messages $U_1, \ldots, U_m$ together produced

[1] This still makes sense since Π is a public-coin protocol; the outside verifier can directly generate a verifier response for any round of the protocol.

a successful completion of h. By definition, the distribution of $v_i^{(j_0)}$ produced by T_{i+1} (i.e., $T^{(2)}$) is just the first message of $U_{j_0}|W^h$. On the other hand, the distribution of $v_i^{(j_0)}$ produced by T_i (i.e., $T^{(3)}$) is just the uniform distribution, just like the first message of U_j.

Since T_{i-1} and T_i only differ in how $v_i^{(j)}$ is produced, their difference in success probability can be bounded by the statistical difference in the distributions of $v_i^{(j)}$. This is in turn bounded by:

$$\sum_{h\in H_i}\sum_{j=1}^{m}\Pr_T[h]\Pr[j_0=j]\Delta(U_j|W^h,U_j)=\sum_{h\in H_i}\Pr_T[h]\left(\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W^h,U_j)\right)(*)$$

Lemma 6 states that for any event W,

$$\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W,U_j)\le\sqrt{\frac{1}{m}\log\left(\frac{1}{\Pr[W]}\right)}$$

Observe that before iteration i, T_i and T_{i+1} are identical to $T^{(2)}$. When $T^{(2)}$ does not abort, $T^{(2)}$ is identical to $T^{(1)}$. In that case, Lemma 3 along with the Markov inequality implies that except with probability $1/(10k)$, $T^{(2)}$ fixes a "good" h with $\mathbb{E}_T[R_i \mid h]\le 30kn^d$, so that

$$\Pr[W^h]=\Pr_S[A^h \mid h]=\frac{1}{\mathbb{E}_T[R_i \mid h]}\ge\frac{1}{30kn^d}$$

We can now break the sum in (*) into two parts. Observe that

$$\sum_{\text{bad } h\in H_i}\Pr_T[h]\left(\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W^h,U_j)\right)\le\sum_{\text{bad } h\in H_i}\Pr_T[h]\le\frac{1}{10k}$$

since statistical distances are bounded by 1, and

$$\sum_{\text{good } h\in H_i}\Pr_T[h]\left(\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W^h,U_j)\right)$$
$$\le\sum_{\text{good } h\in H_i}\Pr_T[h]\sqrt{\frac{1}{m}\log(30kn^d)}\le\sqrt{\frac{1}{m}\log(30kn^d)}$$

since $\sum_{h\in H_i}\Pr_T[h]=1$. Together, they show that (*) is at most

$$\frac{1}{10k}+\sqrt{\frac{1}{m}\log\left(30kn^d\right)}=\frac{1}{10k}+O\left(\frac{1}{k\log n}\right)$$

since $m\ge(k^2\log k)\log^2 n$. Summing up over the hybrids, and recalling that $T^{(2)}$ fails with probability at most $1/5$ (Lemma 5), $T^{(3)}$ fails with probability at most

$$\frac{1}{5}+k\left(\frac{1}{10k}+O\left(\frac{1}{k\log n}\right)\right)\le\frac{3}{10}+O\left(\frac{1}{\log n}\right)$$

□

Lemma 7 shows that T is successful with probability $> 1/2$, and completes the proof of Theorem 2. □

4 Bounded Concurrent Zero-Knowledge

In this section we give a family of public-coin proofs for NP, BoundedConcZK, parametrized by k, assuming the existence of one-way functions. The proof with parameter k has $2k^3 + 4$ rounds, and is k-bounded concurrent zero-knowledge whenever $k = \omega(\log n)$ where n is the input size.

4.1 A Bounded Concurrent Public-Coin ZK Protocol

Our construction of BoundedConcZK is similar in spirit to the concurrent zero-knowledge protocol of [RK99]. Given a language $L \in \mathsf{NP}$ and a parameter k, we construct a two stage public-coin proof $\langle P, V\rangle$ as follows. In stage one, $2k^3$ rounds of messages are exchanged where in each round, the prover gives a statistically binding commitment ([Nao91, HILL99]) of a random bit p_i, and the verifier responds with a random bit v_i; we call $p_i = v_i$ a **correct guess**. In stage two, $\langle P, V\rangle$ runs a 4-round public-coin witness indistinguishable proof of the modified NP statement "either $x \in L$ or that $p_i = v_i$ for $k^3 + k^2/2$ values of i", where x is the problem instance. This can be done, for example, by k parallel repetitions of the GMW 3-coloring protocol [GMW91]. The verifier accepts if the prover is successful with the stage two proof.

Protocol BoundedConcZK

Common Input: An instance x of a language $L \in \mathsf{NP}$ and a parameter k.
Stage One: For i from 1 to $2k^3$:
  $P \to V$: Commit to a random bit p_i.
  $V \to P$: Reply with a random bit v_i.
Stage Two: Run k parallel repetitions of the GMW 3-coloring protocol for the NP statement:

$$\left(\text{there exists distinct } i_1, \ldots, i_{k^3+\frac{1}{2}k^2} \text{ s.t. } p_{i_j} = v_{i_j} \text{ for all } j\right) \vee (x \in L)$$

Fig. 1. Our public-coin black-box bounded concurrent zero-knowledge protocol

We set the round complexity of BoundedConcZK to $O(k^3)$ for the following two reasons. First, by the Chernoff bound, we expect that no adversarial prover can have more than $k^3 + O(\sqrt{k^3})$ correct guesses. Hence BoundedConcZK is sound. On the other hand, a zero-knowledge simulator can repeatedly rewind the verifier until it gets a correct guess. Intuitively (and shown formally later), in each round of stage one, the simulator can set one extra $p_i = v_i$ for *some* session, in addition to "natural luck" (that gives correct guesses for half of the sessions). Since the number of sessions is bounded by k, the simulator is able to have $k^3 + O(k^3/k) = k^3 + O(k^2)$ correct guesses per session. This provides the simulator with a trapdoor to simulate stage two of the protocol, and hence BoundedConcZK is bounded concurrent zero-knowledge. We remark that k^3 was chosen for the sake of simplicity and is not optimized. The formal proof of completeness and soundness can be found in the full version of this paper.

4.2 Black-Box Bounded Concurrent Zero-Knowledge

We construct a black box simulator S such that given a malicious verifier, V^*, S^{V^*} generates the view of V^* in BoundedConcZK, provided that the number of concurrent sessions m satisfies $m \leq k$. The goal of S is to obtain as many correct guesses as possible by rewinding V^*. Towards that goal, S employs a simple greedy strategy to incrementally generate a partial view of V^*. Whenever V^* sends S a first stage message v_i, S checks if it had guessed correctly when committing to p_i. If so, S lengthens the partial view of V^* to include this correct guess. Otherwise, S rewinds V^* back to the previously generated partial view. This "incremental strategy" is somewhat reminiscent of [Lin03], but since our protocol is public-coin, the actual analysis is quite different.

We use superscripts to distinguish messages from different sessions. To prevent S from focusing too much on one particular session, we keep m counters, $c^1, \dots, c^m$, to record how much "work" has been done in each session. In general, S proceeds as follows to incrementally fix the output (originally the empty view is fixed):

1. S commits to a fresh random bit for each stage one prover message.
2. For each stage two proof, S aborts if in this session, $p_i = v_i$ for less than $k^3 + k^2/2$ values of i. Otherwise, S uses this as a witness to complete the stage two proof.
3. If S receives a message v_i^j (from session j) and $c^j < 2k^2$, it checks if the commitment to p_i^j is part of the fixed output. If yes, then nothing can be done, so S simply continues. Otherwise, S checks if $p_i^j = v_i^j$. If yes, S takes the opportunity to fix the execution up to message v_i^j as part of the output and increments c^j; in this case we say v_i^j is *rigged*. If $p_i^j \neq v_i^j$, then S rewinds V^* to start a fresh continuation from the currently fixed output.
4. If S has performed $k-1$ rewinds without rigging a message, and on the kth try again receives $v_i^j \neq p_i^j$ where p_i^j is not fixed and $c^j < 2k^2$, S simply gives up and pretend to *rig* v_i^j anyway (albeit incorrectly). That is, S fixes the output up to message v_i^j and increments c^j.

Claim 8. *S is a k-bounded black-box zero-knowledge simulator when $k \in \omega(\log n)$.*

Proof (sketch). We give a proof sketch here, and defer the full proof to the full version of this paper. Suppose for now that all p_i and v_i are independent and uniformly random (intuitively because the prover commitments are computationally hiding). We claim that except with negligible probability, S will have $k^3 + k^2/2$ correct guesses per session. If so, S can complete the stage two proofs (using the witness indistinguishable property) and generate the view of V^*.

To show our claim, observe that whenever a message is rigged, at most one commitment from each session is fixed to be part of the output, because before a second commitment appears in the same session, S would have tried to rig the first commitment first (unless this session already has $2k^2$ messages rigged). Since S rigs at most $2k^2$ messages from each session, and there are $2k^3$ messages and at most k sessions, every session will actually have $2k^2$ messages rigged.

Since all p_i and v_i are independent, a rigged message is always a correct guess except with probability 2^{-k}. Since there are $m(k^3+2k^2) \leq 2k^4$ messages in total, the union bound says that except with $2k^4 2^{-k}$ probability, all rigged messages are correct guesses. Next, for the $2k^3-2k^2$ messages that are not rigged, we apply the Chernoff bound to see that except with probability $e^{-k/4}$, we should have at least $(k^3 - k^2) - k^2/2$ correct guesses. Thus except with negligible probability, we have a total of $k^3 + k^2/2$ correct guesses as desired. □

Acknowledgements. We would like to thank Johan Håstad and the reviewers for invaluable comments, and for pointing out our work's connection with resettable soundness.

References

[Bar01] Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS 2001, pp. 106–115 (2001)

[BG02] Barak, B., Goldreich, O.: Universal arguments and their applications. In: Computational Complexity, pp. 162–171 (2002)

[BGGL01] Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-sound zero-knowledge and its applications. In: FOCS 2002, pp. 116–125 (2001)

[BIN97] Bellare, M., Impagliazzo, R., Naor, M.: Does parallel repetition lower the error in computationally sound protocols? In: FOCS 1997, pp. 374–383 (1997)

[BL02] Barak, B., Lindell, Y.: Strict polynomial-time in simulation and extraction. In: STOC 2002, pp. 484–493 (2002)

[Blu87] Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, pp. 1444–1451 (1987)

[BM88] Babai, L., Moran, S.: Arthur-merlin games: a randomized proof system, and a hierarchy of complexity class. J. Comput. Syst. Sci. 36(2), 254–276 (1988)

[CG89] Chor, B., Goldreich, O.: On the power of two-point based sampling. J. Complex. 5(1), 96–106 (1989)

[CGGM00] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC 2000, pp. 235–244 (2000)

[CKPR01] Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In: STOC 2001, pp. 570–579 (2001)

[DNS04] Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM 51(6), 851–898 (2004)

[FS90] Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC 1990, pp. 416–426 (1990)

[GK96a] Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology 9(3), 167–189 (1996)

[GK96b] Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SICOMP 25(1), 169–192 (1996)

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SICOMP 18(1), 186–208 (1989)

[GMW91] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM 38(3), 690–728 (1991)

[GO94] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7, 1–32 (1994)

[Gol02] Goldreich, O.: Concurrent zero-knowledge with timing, revisited. In: STOC 2002, pp. 332–340 (2002)

[HILL99] Håstad, J., Impagliazzo, R., Levin, L., Luby, M.: A pseudorandom generator from any one-way function. SICOMP 28, 12–24 (1999)

[Hol07] Holenstein, T.: Parallel repetition: simplifications and the no-signaling case. In: STOC 2007, pp. 411–419 (2007)

[HPPW08] Håstad, J., Pass, R., Pietrzak, K., Wikström, D.: An efficient parallel repetition theorem (2008) (manuscript)

[HRS09] Haitner, I., Rosen, A., Shaltiel, R.: On the (im)possibility of arthur-merlin witness hiding protocols. In: TCC 2009, pp. 220–237 (2009)

[IJK07] Impagliazzo, R., Jaiswal, R., Kabanets, V.: Chernoff-type direct product theorems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 500–516. Springer, Heidelberg (2007)

[IW97] Impagliazzo, R., Wigderson, A.: P = BPP if e requires exponential circuits: Derandomizing the xor lemma. In: STOC 1997, pp. 220–229 (1997)

[Kat08] Katz, J.: Which languages have 4-round zero-knowledge proofs? In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 73–88. Springer, Heidelberg (2008)

[KP01] Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in polylogarithmic rounds. In: STOC 2001, pp. 560–569 (2001)

[KPR98] Kilian, J., Petrank, E., Rackoff, C.: Lower bounds for zero knowledge on the internet. In: FOCS 1998, pp. 484–492 (1998)

[Lin03] Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC 2003, pp. 683–692 (2003)

[Nao91] Naor, M.: Bit commitment using pseudorandomness. Journal of Cryptology 4, 151–158 (1991)

[PRS02] Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS 2002, pp. 366–375 (2002)

[PV07] Pass, R., Venkitasubramaniam, M.: An efficient parallel repetition theorem for arthur-merlin games. In: STOC 2007, pp. 420–429 (2007)

[Raz98] Raz, R.: A parallel repetition theorem. SICOMP 27(3), 763–803 (1998)

[RK99] Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–432. Springer, Heidelberg (1999)

[Ros00] Rosen, A.: A note on the round-complexity of concurrent zero-knowledge. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 451–468. Springer, Heidelberg (2000)

On the Amortized Complexity of Zero-Knowledge Protocols

Ronald Cramer[⋆] and Ivan Damgård[⋆⋆]

CWI/Leiden University and University of Aarhus

Abstract. We propose a general technique that allows improving the complexity of zero-knowledge protocols for a large class of problems where previously the best known solution was a simple cut-and-choose style protocol, i.e., where the size of a proof for problem instance x and error probability 2^{-n} was $O(|x|n)$ bits. By using our technique to prove n instances simultaneously, we can bring down the proof size per instance to $O(|x|+n)$ bits for the same error probability while using no computational assumptions. Examples where our technique applies include proofs for quadratic residuosity, proofs of subgroup membership and knowledge of discrete logarithms in groups of unknown order, and proofs of plaintext knowledge for various types of homomorphic encryptions schemes. The generality of our method stems from a somewhat surprising application of black-box secret sharing schemes.

1 Introduction

In a zero-knowledge protocol, a prover tries to convince a skeptical verifier that a certain statement is true. Except with a small error probability, the verifier should be convinced if and only the statement is indeed true, but should learn nothing beyond the validity of the assertion. The statement can take the form of claiming that the input string x is in a given language L (interactive proofs) or claiming that the prover knows a "witness" w such that (x, w) is in some given relation R (interactive proofs of knowledge).

Zero-knowledge was introduced in [10], and has been the subject of intense research ever since. Zero-knowlegde protocols are interesting as theoretical objects in their own right, but are also very useful as building blocks in larger protocols.

The efficiency of zero-knowledge proofs have been studied in many works, and in some cases, extremely efficient zero-knowledge proofs have been found. For NP complete problems such as Boolean circuit satisfiability Ishai et al. [12] show protocols where the proof size (communcation complexity) is $O(|x| + poly(k))$ where k is a security parameter and $|x|$ is the size of the input x. In prime order groups, Schnorr's protocol[13] proves knowledge of a discrete logarithm using a (honest verifier) zero-knowledge proof of size $O(|x| + n)$ for an error probability

⋆ Supported by NWO VICI.

⋆⋆ Supported by the Danish Strategic Research Council.

of 2^{-n}. In a practical application, one must expect to have to communicate x to make the claim in the first place, so this is essentially optimal[1].

However, for several interesting problems, such methods for improved zero-knowledge protocols do not work. This includes, for instance, the very first problem for which a zero-knowledge protocol was suggested, namely quadratic residuosity, where one proves on input x, N that x is a square modulo N (and that the prover knows a square root). The well-known classical protocol for this has error probability 1/2, and one must repeat it n times for an error probability of 2^{-n} so that the proof will be of size $O(|x|n)$. No more efficient solution with unconditional soundness and zero-knowledge was previously known.

The state of affairs is similar for the discrete log problem in groups of unknown order. Say we are given $g, h \in Z_N^*$ for an RSA modulus N, and the prover claims that h is in the group generated by g, and that he knows the discrete logarithm of h base g. The best solution we know for this has error probability 1/2, and again we must repeat the entire protocol to reduce the error. Schnorr's protocol cannot be used here since its proof of soundness requires that the group order is known, and finding the order of Z_N^* is equivalent to factoring N. Even if we were happy with only a proof of membership in the group generated by g, the error probability for known solutions would be 1 divided by the smallest prime factor in the total group order, which is 1/2 for the case of Z_N^*. It should be noted that Fujisaki and Okamoto[8] have shown how to get around these difficulties, but only if we are guaranteed to be working in a subgroup of Z_N^* with only large prime factors in the order, and then only under the strong RSA assumption.

Other examples of a similar nature come from various proposals for homomorphic encryption where one uses subgroups of Z_N^*, often with small prime factors in their order, to make the decryption algorithm efficient[11,6]. In many applications, one needs a zero-knowledge proof that one knows the plaintext for a given ciphertext, and we then have all the same problems as described above.

In this paper, we show a general method that applies to all the problems mentioned above, and allows us to reduce the proof size in the amortized sense: we can give a proof for n instances of the problem simultaneously such that the communication complexity per instance proved is $O(|x| + n)$ for an error probability of 2^{-n}, thus we are as efficient as the best known unconditional protocols for any problem. The technique uses no computational assumptions. In all cases, the computational complexity is also reduced compared to naive repetition of the basic protocol. Here, the most favorable case is discrete log where computation is reduced by a factor n, for quadratic residuosity we gain a factor $\log n$.

We emphasize that for the case of proofs for quadratic residues, what we achieve is different from what can be done using Fiat-Shamir type protocols [9], although they may seem superficially similar to ours: the Fiat-Shamir protocol is also efficient, it takes n quadratic residues as input and has an error probability of 2^{-n}. The difference lies in the type of witness that the prover proves knowledge of. For Fiat-Shamir, the prover only has to know a product of *some*

[1] Although the proof itself could in principle be even smaller if the associated NP-witness is smaller than x.

of the square roots of the input numbers. In fact, the prover could know nothing about the first input number and still survive with probability 1/2. In contrast, our protocol guarantees that the prover knows all the square roots. It should also be noted that the construction of zero-knowledge protocols from multiparty computation in [12] can be adapted to give a protocol for quadratic residuosity with complexity similar to ours. This requires a computational assumption so that, unlike our work, either zero-knowledge or soundness will only be computational. For the other problems we handle in this work, the construction from [12] either leads to larger complexity than ours, or does not not seem to apply at all.

We give an abstract framework characterizing the type of problem that our method applies to, and derive all the examples above as special cases. Basically, we need a function with certain homomorphic properties on some Abelian groups and a ring A that acts on the groups in a sufficiently nice way. The generality of our method comes from a somewhat surprising application of a new result we show for Black-Box Secret-Sharing. This result allows us to use the integers Z as the ring A, and since any Abelian group is a Z-module, we immediately get a general result.

Applications of our result include multiparty computation based on homomorphic encryption, where players would supply inputs by sending them in encrypted form. To make the overall protocol secure, players must prove that they know the inputs they supply, and our method can be used to give such proofs efficiently for a large number of ciphertexts. Note that some computations, such as certain auctions, do in fact require players to submit large amounts of data as input. Another application involves proofs of negative statements such as proving that a number x is not a square modulo N. The classical protocol for this from [10] uses the proof for quadratic residuosity as a subroutine and has complexity $O(|x|n^2)$. Our method reduces this to $O(|x|n)$ without making any computational assumptions. The same idea can used to prove for some homomorphic encryption schemes that a ciphertext contains a non-zero plaintext. Note that these application are for a single instance proof and so are not of an amortized nature.

Finally, we note that our construction generally leads to protocols that are only honest-verifier zero-knowledge(HVZK), as is the case for Schnorr's protocol. But in this paper we are generally happy with this property since first, it is often sufficient when using a protocol as a building block in a larger construction, and second there are several general techniques that can build zero-knowledge protocols from HVZK ones without significant loss of efficiency. While these techniques typically require a complexity assumption, we believe that the technique by Cramer, Damgård and MacKenzie [2] may lead to a solution with no assumptions, this will be the subject of an upcoming paper.

2 The Basic Idea

The first zero-knowledge proof ever presented was the well known protocol to prove quadratic residuosity. We show here a variant related to Goldwasser-Micali probabilistic cryptosystem, which we will use as a running example in the following. In this cryptosystem, the public key is an RSA modulus N, and we assume for simplicity that it is chosen such that -1 is not a square modulo N. To encrypt a bit w with randomness s, we compute $E_N(w,s) = (-1)^w s^2 \bmod N$. The encryption function is homomorphic, that is, it has two properties: first $E_N(w,s)E_N(w',s') \bmod N = E_N(w \oplus w', ss' \bmod N)$, and moreover, we can multiply a known plaintext b "into a ciphertext", i.e., we have $E_N(w,s)^b = E_N(wb, s^b)$.

Now consider a scenario where the common input to prover P and verifier V is a pair of numbers N and ciphertext x. Now P claims to know a bit w and $s \in Z_N^*$ such that $x = E_N(w,s)$. The protocol goes as follows:

1. P chooses $r \in \{0,1\}, u \in Z_N^*$ at random and sends $a = E_N(r,u)$ to V.
2. V chooses a bit b at random and sends it to P.
3. P sends $z = r \oplus bw, v = us^b \bmod N$ to V, who accepts if and only if $E_n(z,v) = ax^b \bmod N$, and u,v are in Z_N^*.

It is well known that this protocol is perfect zero-knowledge and has error probability 1/2. The reader can easily verify that completeness, soundness and zero-knowledge of the protocol can be based only on the above homomorphic properties of E_N. While error probability of 1/2 is not sufficient in practice, repeating the protocol n times reduces the error probability to 2^{-n}. However, the size of the entire proof will be roughly n times the size of the problem instance.

In this paper we will be concerned with doing it more efficiently if we are to give a proof for n instances of a problem simultaneously. So say we are given a vector $\mathbf{x} = (x_1, ..., x_n)$ of ciphertexts. If we expand the encryption function in a natural way to vectors by applying it to every entry, we can say that the prover's claim now is that he knows vectors $\mathbf{w}, \mathbf{s}$ such that $E_N(\mathbf{w}, \mathbf{s}) = \mathbf{x}$.

Now, the key idea is to consider $\mathbf{w}$, not just as a bit string, but as *an element in the extension field* $GF(2^n)$. Since addition in $GF(2^n)$ is coordinate-wise xor, the (expanded) encryption function is still homomorphic. We have

$$E_N(\mathbf{w}, \mathbf{s})E_N(\mathbf{w}', \mathbf{s}') = E_N(\mathbf{w} + \mathbf{w}', \mathbf{s}\mathbf{s}'),$$

where $\mathbf{w}+\mathbf{w}'$ is addition in $GF(2^n)$ and $\mathbf{s}'\mathbf{s}$ is multiplication in the direct product $(Z_N^*)^n$. We are also able to multiply an element $e \in GF(2^n)$ "into a ciphertext". We can do this by noticing that if we consider $GF(2^n)$ as a vector space over $GF(2)$, multiplication by e is a linear mapping. Taking E to be the matrix of this mapping, multiplying E on an n-bit vector implements multiplication by e. Using this, we can define $\mathbf{x}^e \in (Z_N^*)^n$, where $\mathbf{x} \in (\mathbf{Z_N^*})^{\mathbf{n}}$, namely the i'th entry in $\mathbf{x}^e$ is

$$(\mathbf{x}^e)_i = \prod_{j=1}^{n} x_j^{E(i,j)} \bmod N,$$

where $E(i,j)$ is interpreted as a 0/1 integer. The reader can easily verify that this gives us:

$$E_N(\mathbf{w}, \mathbf{s})^e = E_N(e\mathbf{w}, \mathbf{s}^e).$$

The upshot of this is that since E_N satisfies the same homomorphic properties as before, when seen as a function for encrypting elements in $GF(2^n)$, we can do a proof of knowledge for plaintexts in $GF(2^n)$ by mimicking the protocol above:

1. P chooses $\mathbf{r} \in \{0,1\}^n, \mathbf{u} \in (Z_N^*)^n$ at random and sends $\mathbf{a} = E_N(\mathbf{r}, \mathbf{u})$ to V.
2. V chooses $e \in GF(2^n)$ at random and sends it to P.
3. P sends $\mathbf{z} = \mathbf{r} + e\mathbf{w}, \mathbf{v} = \mathbf{u}\mathbf{w}^e$ to V, who accepts if and only if $E_n(\mathbf{z}, \mathbf{v}) = \mathbf{a}\mathbf{x}^e$, and all entries in $\mathbf{u}, \mathbf{v}$ are in Z_N^*.

Note that V now chooses between 2^n challenges. In fact one can show that if the prover could answer correctly two different challenges e, e', then from the answers we could efficiently compute valid $\mathbf{w}, \mathbf{s}$. The key reason why this is possible is that $e - e'$ is invertible because $GF(2^n)$ is a field (a detailed proof follows as a special case of the general framework we present below).

Hence this protocol has error probability 2^{-n}. Note, however, that we only send a constant number of "compound" ciphertexts to do the protocol. Hence, compared to iterating the basic protocol n times for all n instances which would be the naive solution, we have saved a factor n in the size of the proof.

3 A Framework

In this section we show that the idea we just outlined is not tied to encryption functions over finite fields. All we really need is a function with certain homomorphic properties on Abelian groups, and a ring that acts in "nice" way on the involved groups. To help understand the framework, we use as running example the protocol from the previous section, and show how it is a special case.

3.1 Set-Up and Assumptions

Consider a function $f : R \times S \rightarrow X$, where R, S, X are finite Abelian groups. To make the framework fit with the example instantiations to follow, we will write R additively and S, X multiplicatively.

In what follows, we will always assume that we can sample efficiently from all groups that occur, and compute the group operation and inverses efficiently. We also assume that elements can be communicated in some representation such that membership in the relevant group can be checked efficiently. We assume f is "almost" homomorphic, namely it satisfies the following:

$$f(r,s) \cdot f(r',s') = f(r+r, ss'\delta(r,r')) \text{ and } f(0,s)^{-1} = f(0,s^{-1}) \qquad (1)$$

for all $r, r' \in R, s, s' \in S$ and where $\delta(r, r') \in S$ can be efficiently computed from r, r'.

To connect the framework to the previous example, one may think of $R = Z_2, S = X = Z_N^*$ and $f(r,s) = (-1)^r s^2 \bmod N$, where N is such that -1 is a non-square modulo N. Here, of course, we would have $\delta(r,r') = 1$.

We now assume a commutative ring A with 1 such that R is an A- module (this will be the case if $A = Z$, for instance). We assume that A acts on elements in X, i.e., given $a \in A, x \in X$ one can efficiently compute a new element $x^a \in X$. In the running example, we will set $A = GF(2)$, so an element $a \in A$ is 0 or 1. We then think of these as the *integers* 0 or 1, in which case $x^a \in Z_N^*$ is well defined.

We assume that the action of A on X respects the structure of A, X to some extent, more precisely, we assume for all $x, y \in X, a, b \in A$ that

$$x^a y^a = (xy)^a, \quad x^0 = 1, \quad x^1 = x, \quad 1^a = 1 \tag{2}$$

To complete this picture, we also need an assumption on expressions of form $x^a x^b, (x^a)^b$:

$$x^a x^b = x^{a+b} f(0, \varDelta), \quad (x^a)^b = x^{ab} f(0, \varGamma) \tag{3}$$

for all $x \in X, a, b \in A$, and where $\varDelta, \varGamma$ can be efficiently computed from x, a, b.

For our running example, (2) is trivially satisfied. For (3), one has to remember that the addition in the exponent is in $GF(2)$ and so is actually an xor. Therefore, the first conditions is satisfied if we set $\varDelta = x$ when $a = b = 1$ and $\varDelta = 1$ otherwise. The second condition is satisfied by setting $\varGamma = 1$ always.

We also make an assumption on the way $a \in A$ acts on elements in $Im(f) \subset X$:

$$f(r, s)^a = f(a \cdot r, a(s)) \tag{4}$$

for all $a \in A, r \in R, s \in S$. We make no specific assumptions on $a(s) \in S$, other than it can be computed efficiently from a, s. In our example, (4) is satisfied if we just set $a(s) = s^a \bmod N$ where, as above, we think of a as an integer in the natural way.

In the following, we will consider the direct products A^n, R^n, S^n, X^n for a natural number n. Our final assumption is that there exist a special subset $\varOmega_n \subset A^n$ and an efficiently computable mapping ω which for every $e \in \varOmega_n$ we outputs a matrix $\omega(e)$ with m rows and n columns and entries in A, where m is some function of n and furthermore for every pair $e, e' \in \varOmega_n$ where $e \neq e'$, the matrix $\omega(e) - \omega(e')$ is invertible, i.e., there exists an n by m matrix N such that $N(\omega(e) - \omega(e')) = I_n$. Values $e \in \varOmega_n$ will be used a challenges in our protocols to follow, and since the error probability will be $1/|\varOmega_n|$, we will be looking for constructions that give us a large $\varOmega_n$, preferably of size exponentially large in n. In the following, we will usually use E as shorthand for $\omega(e)$.

Definition 1. *If f, A and ω satisfy all of the above conditions, we say that f is* ZK-friendly with respect to *A and ω.*

In our example, we can set $\varOmega_n$ to be all of $A^n = GF(2)^n$ and $m = n$. Then for $e \in \varOmega_n$, we let $E = \omega(e)$ be the matrix that implements multiplication by e in the field $GF(2^n)$, as in the previous section.

3.2 Notation

We will use $\mathbf{r}, \mathbf{s}$ to denote column vectors of elements in R, respectively S, and $f(\mathbf{r}, \mathbf{s})$ to denote the result of applying f to each coordinate.

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ \cdots \\ s_n \end{pmatrix} \quad f(\mathbf{r}, \mathbf{s}) = \begin{pmatrix} f(r_1, s_1) \\ f(r_2, s_2) \\ \cdots \\ f(r_n, s_n) \end{pmatrix}$$

Let $\mathbf{x}$ be a vector of elements in X, and M is a matrix with entries in A and m rows and n columns. Then we define:

$$\mathbf{x}^M = \begin{pmatrix} \prod_{i=1}^n x_i^{M[1,i]} \\ \prod_{i=1}^n x_i^{M[2,i]} \\ \cdots \\ \prod_{i=1}^n x_i^{M[m,i]} \end{pmatrix} \quad M(\mathbf{s}) = \begin{pmatrix} \prod_{i=1}^n M[1,i](s_i) \\ \prod_{i=1}^n M[2,i](s_i) \\ \cdots \\ \prod_{i=1}^n M[m,i](s_i) \end{pmatrix}$$

It is straightforward to verify that our assumptions on the action of A on X imply that

$$\mathbf{x}^B \mathbf{x}^C = \mathbf{x}^{B+C} f(0^n, \mathbf{\Delta}), \quad (\mathbf{x}^M)^N = \mathbf{x}^{NM} f(0^n, \mathbf{\Gamma}),$$

$$f(\mathbf{r}, \mathbf{s}) f(\mathbf{r}', \mathbf{s}') = f(\mathbf{r} + \mathbf{r}', \mathbf{s}\mathbf{s}' \delta(\mathbf{r}, \mathbf{r}')), \quad f(\mathbf{r}, \mathbf{s})^M = f(M\mathbf{r}, M(\mathbf{r}, \mathbf{s}))$$

for matrices B, C, M, N, where vectors $\mathbf{\Gamma}, \mathbf{\Delta}, M(\mathbf{r}, \mathbf{s})$ can be efficiently computed from the inputs to the operations, and where the function $\delta : R^n \times R^n \to S^n$ is derived from the original δ in the natural way. Note that 0^n denotes the column vector with n zero-entries.

To compute what $M(\mathbf{r}, \mathbf{s})$ should be, one starts from the fact that $(f(\mathbf{r}, \mathbf{s})^M)_j = \prod_{i=1}^n f(\mathbf{r}, \mathbf{s})_i^{M[j,i]}$ and then use (1) and (2). If f is not 1-1, it may be possible to get different values for $M(\mathbf{r}, \mathbf{s})$ depending on the order in which we compute the product. But this is not a problem, in the following we only ned that we can compute *some* element in $M(\mathbf{r}, \mathbf{s}) \in S^n$ that makes $f(\mathbf{r}, \mathbf{s})^M = f(M\mathbf{r}, M(\mathbf{r}, \mathbf{s}))$ be true.

3.3 Some Σ-Protocols

In this section, we assume throughout that we are given a function f that is ZK-friendly w.r.t. some A, ω, and then show that we can build a number of zero-knowledge protocols, more specifically they will be so-called *Σ-protocols.* A *Σ-protocol for a relation* $R = \{(x, w)\}$ is a 3-move protocol for prover P and verifier V. x is the common input and P gets w as private input. Conversations in the protocol have form (a, e, z) where e is a random challenge sent by V. The standard properties of a Σ-protocol is that it is perfectly complete, honest verifier zero-knowledge and sound in the particular sense that from x and conversations $(a, e, z), (a, e', z')$ where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$. This implies that the protocol is a proof of knowledge for R according to the standard definition, with knowledge error 1 divided by the number of possible challenges.

The homomorphic property of f described above already implies that there is a Σ-protocol with error probability $1/2$ for the relation $R = \{(x,(w,s))|\ f(w,s) = x\}$. Namely P sends $a = f(r,u)$ for random r,u, and V asks P to send a preimage of either a or xa. This will called *Protocol 0* in the following.

We now give a Σ-protocol for a set of n instances, where the public input is $\mathbf{x} \in X^n$, and the prover demonstrates knowledge of $\mathbf{w}, \mathbf{s}$ such that $f(\mathbf{w},\mathbf{s}) = \mathbf{x}$. In other words, a Σ-protocol for the relation $R_f = \{(\mathbf{x},(\mathbf{w},\mathbf{s}))|\ f(\mathbf{w},\mathbf{s}) = \mathbf{x}\}$ The protocol works as follows:

Protocol 1

1. P chooses vectors $\mathbf{r}, \mathbf{u}$ of length m at random and sends $\mathbf{a} = f(\mathbf{r},\mathbf{u})$ to V.
2. V selects a random element $e \in \Omega_n$ and sends it to P.
3. P sends $\mathbf{z} = E\mathbf{w} + \mathbf{r}$ and $\mathbf{v} = E(\mathbf{w},\mathbf{s}) \cdot \mathbf{u} \cdot \delta(E\mathbf{w},\mathbf{r})$ to V.
4. V accepts if and only if $f(\mathbf{z},\mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$.

In this protocol, as well as in all the following, the verifier should also check that every communicated group element is in the group it should be in. For the example from the introduction, this translates to checking that numbers communicated are relatively prime to the modulus N.

Lemma 1. *Protocol 1 is a Σ-protocol for R_f, with error probability $1/|\Omega_n|$. The protocol is also an interactive proof that each entry in $\mathbf{x}$ is in $Im(f)$.*

Proof. Completeness is trivial by the homomorphic property of f. For special soundness, we can assume that we have conversations

$$(\mathbf{a}, e, \mathbf{z}, \mathbf{v}), (\mathbf{a}, e', \mathbf{z}', \mathbf{v}'),\ \text{such that}\ f(\mathbf{z},\mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a},\ \mathbf{f}(\mathbf{z}',\mathbf{v}') = \mathbf{x}^{\mathbf{E}'} \cdot \mathbf{a}$$

and we must compute a valid witness for $\mathbf{x}$. Dividing one equation by the other and using our assumptions a few times, we can conclude that

$$f(\mathbf{z} - \mathbf{z}', \mathbf{v} \cdot \mathbf{v}'^{-1}) = \mathbf{x}^{E-E'} \cdot f(0^n, \mathbf{\Delta})$$

for some $\mathbf{\Delta}$ we can compute efficiently. Setting $A = E - E'$ and moving $f(0^n, \mathbf{\Delta})$ to the other side, we see that we can efficiently compute $\mathbf{c}, \mathbf{d}$ and invertible A such that

$$f(\mathbf{c},\mathbf{d}) = \mathbf{x}^A$$

We then apply the inverse N on both sides, and get

$$f(N \cdot \mathbf{c}, N(\mathbf{c},\mathbf{d})) = (\mathbf{x}^A)^N = \mathbf{x} \cdot f(0^n, \mathbf{\Gamma})$$

for an easily computable vector $\mathbf{\Gamma}$. Moving $f(0^n, \mathbf{\Gamma})$ to the other side, we can easily write $\mathbf{x}$ as $f(\mathbf{r},\mathbf{s})$ for known $\mathbf{r},\mathbf{s}$, and so we have the required witness. Since we always obtain something in the preimage of x under f, soundness as a proof of membership follows as well.

Finally, we have to provide an honest verifier simulator. For this, we simply choose $e, \mathbf{z}, \mathbf{v}$ uniformly in their respective domains and let $\mathbf{a} = f(\mathbf{x}, \mathbf{v}) \cdot (\mathbf{x}^E)^{-1}$. This clearly simulates the real conversations perfectly, since $\mathbf{z}, \mathbf{v}$ are indeed uniform in real conversations, and $\mathbf{a}$ is fixed when given $\mathbf{z}, \mathbf{v}$.

An straightforward specialization of Protocol 1 can be used to show that $\mathbf{x} = f(0^n, \mathbf{s})$:

Protocol 1.5

1. P chooses vector $\mathbf{u}$ of length m at random and sends $\mathbf{a} = f(0^n, \mathbf{u})$ to V.
2. V selects a random element $e \in \Omega_n$ and sends it to P.
3. P sends $\mathbf{v} = E(0^n, \mathbf{s}) \cdot \mathbf{u} \cdot \delta(0^n, 0^n)$ to V.
4. V accepts if and only if $f(0^n, \mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$.

Lemma 2. *Protocol 1.5 is a Σ-protocol for the relation* $\{(\mathbf{x}, \mathbf{s}) |\ f(0^n, \mathbf{s}) = \mathbf{x}\}$.

One immediate generalization of Protocol 1 assumes we have two functions f, g that both satisfy our assumptions for the same A, R, S. We can then build a Σ-protocol for the relation $R_{f,g} = \{(\mathbf{x}, \mathbf{x}', (\mathbf{w}, \mathbf{s}, \mathbf{s}')) |\ f(\mathbf{w}, \mathbf{s}) = \mathbf{x}, g(\mathbf{w}, \mathbf{s}') = \mathbf{x}'\}$, i.e., the demand is that the same $\mathbf{w}$ appears in both preimages. The protocol works as follows:

Protocol 2

1. Start two instances of Protocol 1, using as input $\mathbf{x}$ respectively $\mathbf{x}'$. The prover sends $\mathbf{a}, \mathbf{a}'$, computed using the same value of $\mathbf{r}$ in both instances.
2. The verifier sends one challenge e that the prover uses in both instances to compute the answer.
3. The prover sends $\mathbf{z}, \mathbf{v}, \mathbf{z}', \mathbf{v}'$, and the verifier accepts if and only if $\mathbf{z} = \mathbf{z}'$ and $f(\mathbf{z}, \mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$, $g(\mathbf{z}', \mathbf{v}') = \mathbf{x}'^E \cdot \mathbf{a}'$.

By following through the proof for Protocol 1, one trivially obtains

Lemma 3. *Protocol 2 is a Σ-protocol for $R_{f,g}$, with error probability $1/|\Omega_n|$. The protocol is also an interactive proof that each entry in $\mathbf{x}$ is in $Im(f)$ and each entry in $\mathbf{x}'$ is in $Im(g)$.*

Protocols assuming *R* is a ring. We now show that our framework can also be used to show multiplicative relations among preimages under f. To do this, we need to assume that the (additive) group R is actually a ring, and furthermore that we can define an action of R on X and S such that (2), (3), and (4) are satisfied also if we choose $a, b \in R$.

This allows us to define for $x \in Im(f)$ a function

$$f_x(r, s) := x^r \cdot f(0, s)$$

One sees that f_x is almost a homomorphism in the same sense as f, our assumptions immediately imply that we have $f_x(r, s) f_x(r', s') = f_x(r + r', ss'\delta_x(r, r'))$, for some $\delta_x(r, r')$ that is easy to compute from x, r, r'.

Now, suppose we have given $x, y, z \in X$ where a prover knows a, b, c, s_a, s_b, s_c such that $x = f(a, s_a), y = f(b, s_b), z = f(c, s_c)$ and where furthermore $c = ab$. Following several previous works, we can express the relation a bit differently so that it becomes something we can prove using essentially just the protocol we have already.

Notice that if we set $s' = s_c \cdot b(s_a)^{-1} \cdot \delta(ab, 0)^{-1}$, then we have

$$f(c, s_c) = f(ab, s_c) = f_x(b, s')$$

We now consider n instances of such a case, but for a single x and we want a Σ-protocol for the relation R_{mult}, defined as:

$$\{((x, \mathbf{y}, \mathbf{z}), (a, \mathbf{b}, \mathbf{c}, s_a, \mathbf{s_b}, \mathbf{s_c}))|\ x = f(a, s_a), \mathbf{y} = f(\mathbf{b}, \mathbf{s_b}), \mathbf{z} = f(\mathbf{c}, \mathbf{s_c}),\ a \cdot \mathbf{b} = \mathbf{c}\}$$

Then the protocol and lemma below follow immediately:

Protocol 3

1. Run Protocol 0 iterated $\log |\Omega_n|$ times on input x (we can afford to do this on a single input, as it will have the same complexity as the next step).
2. Exploiting the fact that $\mathbf{ab} = \mathbf{c}$, the prover computes $\mathbf{s}'$ as above such that $\mathbf{z} = f_x(\mathbf{b}, \mathbf{s}')$.
3. Do protocol 2 on input $\mathbf{y}, \mathbf{z}$ using f, f_x as the functions f, g.

Lemma 4. *Protocol 3 is a Σ-protocol for R_{mult}.*

As a final example, we show that the framework can be used to show a more negative kind of statement. We need to assume that r is uniquely determined from $f(r, s)$, and second that R is a field. Then we can build an interactive proof system for the language $L = \{x|\ x = f(r, s), r \neq 0\}$.

Protocol 4

1. V chooses n-vectors $\mathbf{r} \in R^n, \mathbf{s} \in S^n$ at random, and computes $\mathbf{g} = f_x(\mathbf{r}, \mathbf{s})$. He sends the $\mathbf{g}$ to P.
2. V uses Protocol 1 to show that he knows $\mathbf{r}, \mathbf{s}$ such that $\mathbf{g} = f_x(\mathbf{r}, \mathbf{s})$.
3. If P accepts the proof in the previous step, he computes $\mathbf{r}$ and sends it to V, who accepts if and only if P sent the correct $\mathbf{r}$

Note that P can do the computation i step 3: since if $x = f(w, s)$ for $w \neq 0$, we have $g_i = x^{r_i} f(0, s_i) = f(wr_i, u_i)$ for some u_i. By assumption wr_i is determined from $f(wr_i, u_i)$ and P can divide out w to get r_i. In general$P$ may need large computing power to find wr_i, but in some cases P can have a trapdoor allowing him do to do it efficiently.

On the other hand if $w = 0$, then $\mathbf{g}$ contains no information on $\mathbf{r}$. Neither does the proof given by V, since it is honest verifier zero-knowledge and hence witness indistinguishable. Therefore, the prover can do no better than a random guess, so the error probability is $|R|^{-n}$. Finally, the protocol is easily seen to be zero-knowledge by a standard argument: the simulator uses rewinding of V to extract $\mathbf{r}$ and can then send exactly what the prover would have sent. If $|R|$ is a small constant such as 2, then Protocol 4 gives a way to improve the complexity over the naive solution where V in step 2 uses Protocol 0 to prove he knows $\mathbf{r}$: we only need to send $O(n)$ group elements, rather than n^2.

3.4 Using Black-Box Secret-Sharing in the Framework

Suppose we are given any function f satisfying (1). Note that if we choose $A = Z$, most of the conditions conditions are automatically satisfied, because any Abelian group is a Z-module. In more concrete terms, it always makes sense to multiply a group element by an integer if the group is written additively (or raise it to an integral power if it is written multiplicatively). In fact, one can easily verify that (2), (3) and (4) are always true if we set $A = Z$, so the only missing condition is the existence of the special subset Ω_n in Z^n and the mapping ω.

A construction of such a set follows from the black-box secret-sharing scheme we introduce in the full version of this paper [1]. The construction itself, however, is easy to understand without this background: recall that Ω_n must be a subset of $A^n = Z^n$. We choose Ω_n to be the set of vectors with entries that are 0 or 1, thus Ω_n has size 2^n. We then need to build, from $e \in \Omega_n$, a matrix $\omega(e)$ with n columns and m rows, where we choose $m = 2n - 1$, and where $e \neq e'$ implies that $\omega(e) - \omega(e')$ is invertible. We do this as follows: thinking of e as a column vector, the j'th column of $\omega(e)$ starts with $j - 1$ zeros, followed by e, followed by $n - j$ zeros.

It is straightforward to show that for any two different e, e', indeed $\omega(e) - \omega(e')$ has an inverse N such that $N(\omega(e) - \omega(e'))$ is the identity matrix. One just observes that the matrix $\omega(e) - \omega(e')$ must always be upper triangular with only 1's or -1's on the diagonal. Therefore we have:

Theorem 1. *Any f satisfying (1) is ZK-friendly with respect to Z and ω constructed as above. In particular, the Σ-protocols 1, 1.5, 2, 3 and 4 will have error probability 2^{-n} and communication complexity linear in n.*

To understand where this construction comes from and why it is connected to secret sharing, it is instructive to have a look at the classical protocol for discrete logarithms, where the prover knows w such that $h = g^w$ in some finite group. The prover sends $a = g^r$, the verifier chooses challenge $e = 0$ or 1, and the prover returns $z = r + ew \bmod t$ where t is the order of g. The verifier checks that $g^z = a \cdot h^e$.

One can interpret this protocol as being based on a very simple 2 out of 2 secret sharing scheme, where the secret is w, r is the randomness used for the sharing, and the shares are r and $r + w$. In this language, the protocol is that the prover commits to the randomness for the secret sharing by sending $a = g^r$, and must then reveal the share of the verifiers choice. The verifier's check ensures that the correct share is indeed revealed. On one hand, since 2 shares are enough to reconstruct, we can extract the secret from any prover who can answer 2 different challenges. On the other hand, since one share reveals no information on the secret, we can simulate the protocol without knowing the secret.

If the group order t is public and is a prime, we can instead use the obvious linear 2 out of t secret sharing scheme where there are t shares and the e'th share is $r + ew \bmod t$. If we again build a protocol by asking the prover to commit to the randomness by sending g^r and then reveal the share of the verifier's choice, we get exactly Schnorr's protocol. From this point of view, the efficiency of this protocol can be explained from the fact that it is based on a 2 out of t secret sharing scheme for a very large t.

Our protocols from the previous section can be interpreted in a similar way, and we if combine this with the idea of using $A = Z$, we can rephrase our goal as follows: our protocols work with secrets that are vectors of elements in some Abelian group. What we want is to construct a 2 out of T secret sharing scheme (where T can hopefully be chosen very large) which works by acting on the secret vector by integer matrices, and where shares are vectors that are hopefully not much longer than the secret vector. Moreover the scheme should work for any Abelian group. What we are asking for is in fact a novel black-box secret sharing scheme, a concept which is explained in the full version of this paper, where we also develop the secret-sharing scheme that underlies the above theorem.

4 Examples

4.1 Quadratic Residuosity

Let N be a composite number, and let y be a non-square mod N. Then we can set $R = GF(2), S = X = Z_N^*, f(r,s) = y^r s^2 \bmod N, A = GF(2)$.

Now, we can let vectors in $A^n = GF(2)^n$ correspond in the standard way to elements in the extension field $GF(2^n)$. Multiplication by an element $e \in GF(2^n)$ is a linear mapping, so we set $m = n$ and let E be the matrix of this mapping. Finally we can set Ω_n to be all of $GF(2)^n$ since any non-zero element in $GF(2^n)$ is invertible. It is straightforward to check that this satisfies all our assumptions in the framework. Protocol 1 above now becomes a proof that the prover knows how to decrypt n ciphertexts in the Goldwasser-Micali cryptosystem.

The computational cost of the protocols are clearly dominated by the cost of computing the action of E on the vector $\mathbf{x}$. Doing this is equivalent to computing n products of various subsets of n given elements in Z_n^*. Using a straightforward variant of the so called 4 Russians algorithm, this can be done using $O(n^2/\log n)$ multiplications modulo N. We therefore have:

Corollary 1. *Protocol 1 instantiated for the quadratic residuosity case is a proof that the prover knows how to decrypt n ciphertexts in the Goldwasser-Micali cryptosystem. It has communication complexity $2n$ elements in Z_N^* plus $2n$ bits, error probability 2^{-n}, and the computational complexity is $O(n^2/\log n)$ multiplications modulo N.*

Note that if we wanted to obtain the same error probability using simple repetition of the standard cut-and-choose protocol, the cost for all n instances would be $2n^2$ group elements plus $2n$ bits and the computational cost $O(n^2)$ multiplications modulo N. Protocol 1.5 instantiated for this case is easily seen to be a proof that n input numbers are all squares modulo N. It may seem that to use this protocol we need that a non-square y is given, to define the function f, but this is not the case, since we only need to evaluate f on inputs where the first component is 0, and we always have $f(0, s) = s^2 \bmod N$ no matter which y we would use.

Protocol 4 instantiated for this case is a proof that a given number is a non-square modulo N and this improves the complexity of the classical protocol for this problem from [10] by a factor of n. Again, one can verify that we do not need a non-square y given a priori.

Finally, Protocol 3 in this case becomes a protocol proving that encrypted bit a and encrypted bitstrings $\mathbf{b}, \mathbf{c}$ satisfy $a \wedge \mathbf{b} = \mathbf{c}$, where $a \wedge \mathbf{b}$ is the string obtained by taking the and of a and each bit in $\mathbf{b}$.

4.2 Discrete Log in a Group of Unknown Order

Let N be an arbitrary natural number and $g \in Z_N^*$. Then we will set $R = Z$, S to be the trivial group with one element, and $X = Z_N^*$. We then let $f(r, 1) = g^r \bmod N$. We also set $A = Z$. This does not quite satisfy our framework, since R is not finite, but we will fix this shortly.

The construction behind Theorem 1 implies that we can satisfy the conditions in our framework by construct the set Ω_n as the subset of Z^n consisting of binary strings.

In this case, protocol 1 has to be tweaked slightly: instead of choosing $\mathbf{r}$ uniformly in R^n, which does not make sense when R is infinite, we choose the entries as uniform $\log n + 2k$-bit numbers. This choice ensures both that $f(\mathbf{r})$ will be statistically close to uniform in $Im(f)^m$, and that the entries in $\mathbf{z}$ will be statistically close to uniform $\log n + 2k$-bit numbers. This follows from the fact that the entries in $E \cdot \mathbf{w}$ will be at most $\log n + k$-bit numbers.

The protocol now becomes an interactive proof that the input numbers $x_1, .., x_n$ are all in the group generated by g, and it is a proof that the prover knows the discrete logarithms. The protocol will be honest verifier statistical zero-knowledge.

Corollary 2. *Protocol 1 instantiated for the discrete log in Z_n case is an interactive proof that the input numbers $x_1, .., x_n$ are all in the group generated by g, and it is a proof that the prover knows the discrete logarithms. Let k be the bit length of N. Then the communication complexity is $O(kn)$ bits, the error probability 2^{-n}, and the computational complexity is $O(nk + n^2)$ multiplications modulo N.*

If we wanted to obtain error probability 2^{-n} using simple repetition of the standard cut-and-choose protocol, the cost for n instances would be communication $O(n^2 k)$ bits and also $O(n^2 k)$ multiplications modulo N. So we see that if we choose, e.g., $n = k$, our solution saves a factor k in both the communication and computational complexity.

4.3 Homomorphic Encryption

We already mentioned earlier how our technique can be used for the Goldwasser-Micali probabilistic public-key scheme. This generalizes in a very natural way to encryptions schemes based on higher degree residuosity, say degree q for q a prime larger than 2, provided q divides $\phi(N)$. The plaintext-space for the encryption would be $R = Z_q$ and one would then define the encryption of plaintext r as $f(r, s) = y^r s^q \bmod N$ where y is not a q-power modulo N. The basic Protocol 0 with $A = Z_q$ and $\Omega_n = Z_q^n$ gives a proof of knowlegde of the plaintext for a given ciphertext with error probability $1/q$. Using Protocol 1, this can be amplified to a proof for n plaintexts with error probability q^{-n}, at cost n times the cost of Protocol 0.

In [11], a different type of encryption function is proposed, also based on a composite modulus N and two elements $g, h \in Z_N^*$. The encryption function is $f(m, s) = g^m h^s \bmod N$. Here m is the message chosen in Z_M for a public M and s is chosen at random in some interval $[0..T]$. We do not need to go into the details of the scheme and its security here, it is enough to say that the order of h has to be secret and one needs to assume for security that a random element in the group generated by h cannot be efficiently distinguished from a random element in Z_N^*.

Standard methods for proving in zero-knowledge that you know m, s for a given ciphertext have error probability 1/2, namely one does the obvious Σ-protocol with a binary challenge. One cannot do better using Schnorr-like techniques because one would need to know the order of h to do the knowledge extraction required for soundness. However, the scheme fits in our framework, by setting $R = Z_M$, $S = Z$, $X = Z_N^*$ and $A = Z$. Now, using Theorem 1, Protocol 1 shows that we can prove knowledge of n plaintexts with error probability 2^{-n} at cost about $2n$ times the standard protocol for a single instance.

Finally, we note that if g has order M, R can act on S and X as required for Protocols 3 and 4. Protocol 3 can be used to show multiplicative relations among plaintexts, and in case the plaintext space is a field (i.e., if M is a prime). Protocol 4 can be used to show that a ciphertext contains a non-zero plaintext.

References

1. Cramer, R., Damgård, I.B.: On the Amortized Complexity of Zero-Knowledge Protocols. Full version of this paper (in preparation)
2. Cramer, R., Damgård, I.B., MacKenzie, P.D.: Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
3. Cramer, R., Fehr, S.: Optimal Black-box Secret Sharing over Arbitrary Abelian Groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 272. Springer, Heidelberg (2002)
4. Cramer, R., Fehr, S., Stam, M.: Primitive Sets over Number Fields and Black-Box Secret Sharing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 344–360. Springer, Heidelberg (2005)
5. Damgård, I.B., Ishai, Y.: Scalable Secure Multiparty Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)
6. Damgård, I.B., Geisler, M., Krøigaard, M.: Efficient and Secure Comparison for On-Line Auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)

7. Desmedt, Y., Frankel, Y.: Homomorphic Zero-Knowledge Threshold Schemes over Any Finite Abelian Group. SIAM J. on Discrete Mathematics 7(4), 667–679 (1994)
8. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
9. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1985)
10. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: Proceedings of STOC 1985, pp. 291–304 (1985)
11. Groth, J.: Cryptography in Subgroups of Zn. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 50–65. Springer, Heidelberg (2005)
12. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of STOC 2007, pp. 21–30 (2007)
13. Schnorr, C.-P.: Efficient Signature Generation by Smart Cards. J. Cryptology 4(3), 161–174 (1991)

Linear Algebra with Sub-linear Zero-Knowledge Arguments

Jens Groth[⋆]

University College London
j.groth@ucl.ac.uk

Abstract. We suggest practical sub-linear size zero-knowledge arguments for statements involving linear algebra. Given commitments to matrices over a finite field, we give a sub-linear size zero-knowledge argument that one committed matrix is the product of two other committed matrices. We also offer a sub-linear size zero-knowledge argument for a committed matrix being equal to the Hadamard product of two other committed matrices. Armed with these tools we can give many other sub-linear size zero-knowledge arguments, for instance for a committed matrix being upper or lower triangular, a committed matrix being the inverse of another committed matrix, or a committed matrix being a permutation of another committed matrix.

A special case of what can be proved using our techniques is the satisfiability of an arithmetic circuit with N gates. Our arithmetic circuit zero-knowledge argument has a communication complexity of $O(\sqrt{N})$ group elements. We give both a constant round variant and an $O(\log N)$ round variant of our zero-knowledge argument; the latter has a computation complexity of $O(N/\log N)$ exponentiations for the prover and $O(N)$ multiplications for the verifier making it efficient for the prover and very efficient for the verifier. In the case of a binary circuit consisting of NAND-gates we give a zero-knowledge argument of circuit satisfiability with a communication complexity of $O(\sqrt{N})$ group elements and a computation complexity of $O(N)$ multiplications for both the prover and the verifier.

Keywords: Sub-linear size zero-knowledge arguments, public-coin special honest verifier zero-knowledge, Pedersen commitments, linear algebra, circuit satisfiability.

1 Introduction

It has long been known [Kil92] that zero-knowledge arguments (with computational soundness) can have very low communication. However, known examples of communication-efficient zero-knowledge arguments tend to get their efficiency at the cost of increased computational complexity. Obtaining zero-knowledge arguments that are efficient with respect to *both* communication *and* computation is considered one of the important challenges in theoretical computer science [Joh00]. We address this challenge by constructing zero-knowledge arguments for statements related to linear algebra over finite fields that have sub-linear communication and at the same time also have low computational complexity.

[⋆] Part of this research was done while visiting IPAM, UCLA.

1.1 Our Contribution

We consider row vectors of elements from a finite field $\mathbb{Z}_p$, where p is a large prime. Using a generalization of the Pedersen commitment we can commit to a vector of n elements from $\mathbb{Z}_p$. Each commitment consists of a single group element. A set of n commitments, can be considered a commitment to the n rows of an $n \times n$ matrix. This paper is about zero-knowledge arguments for a set of committed vectors and matrices satisfying a set of linear algebra relations, for instance that a committed matrix is the product of two other committed matrices. We give zero-knowledge arguments with a communication complexity of $O(n)$ elements, i.e., the square-root of the size of the matrices. In addition, the arguments are computationally efficient for both the prover and the verifier. The verifier is a public-coin verifier and does not need to take much action until the end of the argument, where the small size of the arguments makes it possible to verify the correctness using only little computation.

Our sub-linear size zero-knowledge arguments work for a wide range of linear algebra relations. We can commit to single field elements, vectors of field elements and square matrices of field elements. Our results also hold for non-square matrices, however, for simplicity we focus just on square matrices here. Given commitments to field elements, vectors and matrices we can prove relations such as a committed field element being the dot product of two committed vectors, a committed matrix being the product of two other committed matrices, or a committed vector being the Hadamard product (the entry-wise product) of two other vectors. Being able to prove such linear algebra relations makes it possible to address many other statements frequently arising in linear algebra. We can for instance prove that committed matrices are upper or lower triangular, have a particular trace, compute the sums of the rows or columns or prove that a committed matrix is the inverse of another committed matrix. We can also permute the entries of a matrix using either a public or a hidden permutation. Using the linear algebra relations, we also get sub-linear size zero-knowledge arguments for the satisfiability of arithmetic circuits and for the satisfiability of binary circuits demonstrating the generality of our results.

1.2 Related Work

Recent work on zero-knowledge proofs [IKOS07] give us proofs with a communication complexity that grows linearly in the size of the statement to be proven and [IKOS07, KR08, GKR08] also give us proofs with size that depend quasi-linearly on the witness-length. If we consider arguments, the communication complexity can be even lower and Kilian [Kil92] gave a zero-knowledge argument for circuit satisfiability with polylogarithmic communication. His argument goes through the PCP-theorem [AS98, ALM$^+$98, Din07] and uses a collision-free hash-function to build a hash-tree that includes the entire PCP though. Even with the best PCP constructions known to date [BSGH$^+$05] Kilian's argument has high computational complexity for practical parameters. In contrast, our goal is to get short zero-knowledge arguments that are simple and efficient enough for both prover and verifier to be used in practice.

Groth and Ishai [GI08] gave a zero-knowledge argument for correctness of a shuffle [Cha81] of N ElGamal ciphertexts. We rely on techniques developed by Groth and

Ishai and as described below also develop several new techniques. For comparison, we believe it would be possible to modify their argument into an argument for circuit satisfiability with a sub-linear communication of $O(N^{2/3})$ group elements, but the corresponding computational complexity for the prover would be a super-linear number of exponentiations.

1.3 Our Techniques

The generality of our results relies on using randomization and batch-verification techniques to reduce linear algebra relations to equations of the form

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i,$$

where $\boldsymbol{x}_i, \boldsymbol{y}_i$ are committed vectors in $\mathbb{Z}_p^n$, z is a committed field element, and $* : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ is a bilinear map. Besides greatly simplifying the task, the bilinear map also helps reduce computation because it maps pairs of n-element vectors into single field elements giving the prover less to commit to.

Groth and Ishai [GI08] gave a sub-linear size public-coin zero-knowledge argument for the correctness of a shuffle of N ElGamal ciphertexts, with a communication complexity of $O(N^{2/3})$ group elements. We use similar techniques but by making more careful use of the public-coin challenges, we can reduce the communication complexity for our zero-knowledge arguments to $O(\sqrt{N})$ elements. The difference from Groth and Ishai's work is that they choose a set of challenges at random, whereas we let the prover process the verifier's challenges to get a more structured set of challenges. This processing consists of taking a challenge $e \in \mathbb{Z}_p$ from the verifier and using it to generate a set of challenges $(1, e, e^2, \ldots)$, which is a row of a Vandermonde matrix. In the zero-knowledge argument, we then arrange the challenges from the Vandermonde vector in such a way that it leads to many terms cancelling out with each other.

Groth and Ishai's shuffle argument suffered from an increase in the prover's computation complexity in comparison with shuffle arguments that do not have sub-linear size. The same effects apply to some extent to our zero-knowledge arguments when using a constant number of rounds, however, by allowing a logarithmic number of rounds we can eliminate the computational overhead. This is of interest in scenarios where round complexity matters less than computation, for instance in cases where the Fiat-Shamir heuristic is used to make the zero-knowledge argument non-interactive by letting the prover use a cryptographic hash-function to compute the verifier's challenges.

2 Preliminaries

Given two functions $f, g : \mathbb{N} \rightarrow [0,1]$ we write $f(\kappa) \approx g(\kappa)$ when $|f(\kappa) - g(\kappa)| = O(\kappa^{-c})$ for every constant c. We say that f is *negligible* when $f(\kappa) \approx 0$ and that it is *overwhelming* when $f(\kappa) \approx 1$.

We write $y = A(x; r)$ when the algorithm A, on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting

$y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S. We write $(e_1, \ldots, e_m) \leftarrow \text{Van}_m(\mathbb{Z}_p)$ when we pick $e \leftarrow \mathbb{Z}_p$ and define $e_1, \ldots, e_m$ by $e_i = e^{i-1} \bmod p$, corresponding to $(e_1, \ldots, e_m)$ being a row of a Vandermonde matrix.

2.1 Zero-Knowledge Arguments of Knowledge

We are interested in zero-knowledge arguments of knowledge for statements involving linear algebra. We define an argument of knowledge as an argument that has witness-extended emulation, which means we can emulate the entire zero-knowledge argument and at the same time extract a witness. For simplicity, we focus on special honest verifier zero-knowledge (SHVZK) arguments in the common reference string model. There is no loss of generality here: by using a coin-flipping protocol the SHVZK arguments can be converted into arguments with full zero-knowledge against a cheating verifier and the cost of this conversion is insignificant [Gro04]. Moreover, the common reference string may be a random string and may even be chosen by the verifier. We refer to the full paper for further discussion and formal definitions.

2.2 Homomorphic Commitments

The central tool in our SHVZK arguments is a homomorphic commitment to n elements in $\mathbb{Z}_p$, where p is a κ-bit prime. Any homomorphic commitment scheme can be used, but for simplicity and for the sake of making a concrete efficiency analysis, we will in this paper use a generalization of Pedersen commitments [Ped91]. This commitment scheme is length-reducing; a commitment is a single group element no matter how large n is. The length-reduction is crucial, by working on short commitments instead of long vectors we get SHVZK arguments with sub-linear communication complexity.

The generalized Pedersen commitment scheme works as follows. The key generation algorithm K generates a commitment key $ck = (G, g_1, \ldots, g_n, h)$, where $g_1, \ldots, g_n, h$ are randomly chosen generators of a group G of prime order p with $|p| = \kappa$. The message space is $\mathbb{Z}_p^n$, the randomizer space is $\mathbb{Z}_p$ and the commitment space is G. We require that G is a group where it is easy to determine membership and compute the binary operations and assume parties check that commitments are valid, by checking $c \in G$.[1]

To commit to a vector $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ we pick randomness $r \leftarrow \mathbb{Z}_p$ and compute the commitment $c = h^r \prod_{i=1}^n g_i^{x_i}$. As a matter of notation we will write $\text{com}_{ck}(\boldsymbol{x}; r)$ when committing to a vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ using randomness r. In some cases we will commit to less than n elements; this can be accomplished quite easily by setting the remaining messages to 0. When committing to a single element $x \in \mathbb{Z}_p$ using randomness r, we write $\text{com}_{ck}(x; r)$. The generalized Pedersen commitment is perfectly hiding

[1] If the commitments belong to a group $\mathbb{Z}_q^*$ batch verification techniques can be used to lower the cost of checking group membership of many commitments. See also [Gro03] for a variant of the Pedersen commitment scheme over $\mathbb{Z}_q^*$ that makes it possible to almost eliminate the cost of verifying validity. If G is an elliptic curve of order p, then the validity check just consists of checking that c is a point on the curve, which is inexpensive.

since no matter what the messages are, the commitment is uniformly distributed in G. The commitment is computationally binding under the discrete logarithm assumption; we will skip the simple proof.

The common reference string in our SHVZK arguments will be a commitment key ck. We remark that for typical choices of the group G, the commitment key can be easily sampled from a common random string and it is easy to verify that ck is a valid commitment key. It may even be chosen by the verifier, provided the prover and verifier use a p for which it is possible to generate groups where the discrete logarithm problem is hard.

The generalized Pedersen commitment is homomorphic. For all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}_p^n$ and $r, r' \in \mathbb{Z}_p$ we have

$$\mathrm{com}_{ck}(\boldsymbol{x}; r) \cdot \mathrm{com}_{ck}(\boldsymbol{x}'; r') = \mathrm{com}_{ck}(\boldsymbol{x} + \boldsymbol{x}'; r + r').$$

Knowledge of content of commitments. There are standard techniques for proving knowledge of the opening of many commitments, see the full paper. This can be done in 3 rounds and costs little in terms of communication and computation. Therefore, we will for simplicity and without loss of generality often assume without explicitly stating it that the prover knows the openings of the commitments that she sends to the verifier.

2.3 Multi-exponentiation Techniques

Multi-exponentiation techniques allow computing products of the form $\prod_{i=1}^{n} g_i^{x_i}$ faster than computing n single exponentiations. Multi-exponentiations appear frequently in the paper, for instance when computing the generalized Pedersen commitment described earlier. Pippenger [Pip80] developed a general theory of multi-exponentiations; we recommend Lim's presentation [Lim00] of concrete multi-exponentiation techniques with a complexity of less than $2n\kappa/\log n$ multiplications in G, when n is large.

3 Equations with Matrices and Vectors

We wish to commit to matrices and vectors of elements from $\mathbb{Z}_p$ and make SHVZK arguments for them satisfying equations commonly arising in linear algebra. We first consider the following 6 types of equations over committed matrices $X_i, Y_i, Z \in \mathrm{Mat}_{n \times n}(\mathbb{Z}_p)$, committed row vectors $\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z} \in \mathbb{Z}_p^n$ and committed elements $z \in \mathbb{Z}_p$, with public $a_i \in \mathbb{Z}_p$.

$$\boldsymbol{z}^\top = \sum_{i=1}^{m} a_i X_i \boldsymbol{y}_i^\top \qquad Z = \textstyle\sum_{i=1}^{m} a_i X_i Y_i \qquad Z = \displaystyle\sum_{i=1}^{m} a_i X_i \circ Y_i$$

$$z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top \qquad \boldsymbol{z} = \textstyle\sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i \qquad \boldsymbol{z} = \displaystyle\sum_{i=1}^{m} a_i \boldsymbol{x}_i \circ \boldsymbol{y}_i,$$

where $\circ$ is the Hadamard product (entry-wise product). In this section, we will show how to reduce a set of such equations to a couple of equations of the form

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i,$$

where $* : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ is a bilinear map. One bilinear map we will use is the standard dot product of vectors $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^\top$. Another bilinear map we will use is given by $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}(\boldsymbol{y} \circ \boldsymbol{t})^\top$, where $\boldsymbol{t} \in \mathbb{Z}_p^n$ is a public vector chosen by the verifier.

The first step in the reduction is very simple. Since we have committed to row vectors, the three types of equations in the top involving matrices X_i are actually just sets of n equations of the types below. We can therefore focus on the three types of equations on the bottom.

3.1 Reducing Many Equations of the Form $z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top$ to a Single Equation

Randomization can be used to reduce Q equations of the form $z_q = \sum_{i=1}^{m_q} a_{qi}\boldsymbol{x}_{qi}\boldsymbol{y}_{qi}^\top$ to one single equation of the form $z = \sum_{i=1}^{m} \boldsymbol{z}_i \boldsymbol{y}_i^\top$, where $m = \sum_{q=1}^{Q} m_q$. The verifier selects $(r_1, \ldots, r_Q) \leftarrow \mathrm{Van}_Q(\mathbb{Z}_p)$ (observe this only requires the verifier to transmit one field element) and require the prover to demonstrate

$$\sum_{q=1}^{Q} r_q z_q = \sum_{q=1}^{Q} \sum_{i=1}^{m_q} (r_q a_{qi} \boldsymbol{x}_{qi}) \boldsymbol{y}_{qi}^\top.$$

This is a comparison of two degree $Q-1$ polynomials in the challenge consisting of a field element. By the Schwartz-Zippel lemma, there is probability at most $\frac{Q-1}{p}$ for the test to pass unless indeed all the equations hold. Setting $z = \sum_{q=1}^{Q} r_q z_q$ and $\boldsymbol{x}'_{qi} = r_q a_{qi} \boldsymbol{x}_{qi}$, whose commitments can easily be computed using the homomorphic property of the commitment scheme, we get the following equation of the desired form

$$z = \sum_{q=1}^{Q} \sum_{i=1}^{m_q} \boldsymbol{x}'_{qi} \boldsymbol{y}_{qi}^\top.$$

3.2 Reducing $z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i$ to the Form $z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top$

We will now give a 3-move reduction of $\boldsymbol{z} = \sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i$ to the form $z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top$. The verifier picks $\boldsymbol{t} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and asks the prover to demonstrate

$$\boldsymbol{z}\boldsymbol{t}^\top = (\sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i)\boldsymbol{t}^\top = \sum_{i=1}^{m} a_i \boldsymbol{x}_i (Y_i \boldsymbol{t}^\top).$$

By the Schwartz-Zippel lemma, there is at most probability $\frac{n-1}{p}$ of this test passing unless indeed the values satisfy the equation. The problem is that the verifier does not have a straightforward way to compute commitments to $Y_i\boldsymbol{t}^\top$ since we have commitments to the rows of the matrices, but here the verifier is asking for a linear combination of the columns. Choosing $\boldsymbol{t}$ and sending it to the prover is therefore only the first round of the reduction; there will be two more rounds.

For each matrix Y_i the prover creates a new commitment to $\boldsymbol{y}_i = \boldsymbol{t}Y_i^\top$ and sends it to the verifier. The equation can now be reduced to the form

$$\boldsymbol{z}\boldsymbol{t}^\top - \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top = 0,$$

which is of the desired form. In the process we have for each matrix Y_i introduced an additional equation $\boldsymbol{y}_i = \boldsymbol{t}Y_i^\top$ that we need to prove too. We pick $\boldsymbol{s} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and ask the prover to demonstrate

$$\boldsymbol{y}_i\boldsymbol{s}^\top = (\boldsymbol{s}Y_i)\boldsymbol{t}^\top.$$

This is the key idea in this reduction, $\boldsymbol{s}Y_i$ is a combination of row vectors from Y_i and thus easily computable. Using the homomorphic properties of the commitment scheme both the prover and the verifier can compute a commitment to $\boldsymbol{s}Y_i$.

We remark that since the last step in this reduction simply consists of the verifier picking a challenge $\boldsymbol{s}$, we can run the last round in parallel with the reduction in Section 3.1, so our reduction only costs 2 additional rounds. Further, we note that for all Y_i in all equations, we can use the same $\boldsymbol{s}$ and $\boldsymbol{t}$. In the randomization step in the reduction in Section 3.1 we can use the homomorphic properties of the commitment scheme to combine all the vectors that we combine with respectively $\boldsymbol{s}$ and $\boldsymbol{t}$. The main cost of the reduction is therefore the computation of the $\boldsymbol{y}_i$'s and the $\boldsymbol{s}Y_i$'s and the commitments to $\boldsymbol{y}_i$, the rest has modest cost.

3.3 Reducing Equations with Hadamard Products to a Single Equation with a Bilinear Map

We will now reduce a set of Q Hadamard equations of the form

$$\boldsymbol{z}_q = \sum_{i=1}^{m_q} a_{qi}\boldsymbol{x}_{qi} \circ \boldsymbol{y}_{qi}$$

to a single equation. The verifier picks $(r_1, \ldots, r_Q) \leftarrow \mathrm{Van}_Q(\mathbb{Z}_p)$ and requires the prover to give an argument for

$$\sum_{q=1}^{Q} r_q\boldsymbol{z}_q = \sum_{q=1}^{Q}\sum_{i=1}^{m_q}(r_q a_{qi}\boldsymbol{x}_{qi}) \circ \boldsymbol{y}_{qi}.$$

Setting $\boldsymbol{x}'_{qi} = r_q a_{qi}\boldsymbol{x}_{qi}$ and $\boldsymbol{z}' = \sum_{q=1}^{Q} r_q\boldsymbol{z}_q$, whose commitments can be computed using the homomorphic properties, this gives us the equation $\boldsymbol{z}' = \sum_{q=1}^{Q}\sum_{i=1}^{m_q}\boldsymbol{x}'_{qi} \circ \boldsymbol{y}_{qi}$.

Consider now a Hadamard equation of the form $\boldsymbol{z} = \sum_{i=1}^{m} \boldsymbol{x}_i \circ \boldsymbol{y}_i$. We can simplify this equation by picking $\boldsymbol{t} \leftarrow \mathrm{Van}_m(\mathbb{Z}_p)$ and requiring the prover to show

$$\boldsymbol{z}\boldsymbol{t}^\top = (\sum_{i=1}^{m} \boldsymbol{x}_i \circ \boldsymbol{y}_i)\boldsymbol{t}^\top = \sum_{i=1}^{m} \boldsymbol{x}_i(\boldsymbol{y}_i \circ \boldsymbol{t})^\top.$$

Defining the bilinear map

$$* : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p \qquad (\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x}(\boldsymbol{y} \circ \boldsymbol{t})^\top,$$

we have reduced the equation to

$$0 = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i - \boldsymbol{z} * \boldsymbol{1}.$$

4 SHVZK Arguments for a Vector Product Equation

We saw in the previous section that equations involving matrices and vectors could be efficiently reduced to an equation of the form

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i,$$

where $*$ is one of the two bilinear maps $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^\top$ or $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}(\boldsymbol{y} \circ \boldsymbol{t})^\top$. In this section we will give a SHVZK argument of knowledge of openings $z \in \mathbb{Z}_p$ and $\boldsymbol{x}_1, \boldsymbol{y}_1, \ldots, \ldots, \boldsymbol{x}_m, \boldsymbol{y}_m \in \mathbb{Z}_p^n$ satisfying such an equation.

4.1 The Minimal Case

We first give a well-known argument for the minimal case $m = 1$. We have three commitments a, b, c to $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^n$ and $z \in \mathbb{Z}_p$ respectively and the prover wants to convince the verifier that $z = \boldsymbol{x}\boldsymbol{y}^\top$. The prover's private input in the argument consists of the openings $(\boldsymbol{x}, r)$, $(\boldsymbol{y}, s)$ and (z, t) of a, b and c respectively.

P→ V: Pick $\boldsymbol{d}_x, \boldsymbol{d}_y \leftarrow \mathbb{Z}_p^n$, $d_z \leftarrow \mathbb{Z}_p$ and randomizers $r_d, s_d, t_1, t_0 \leftarrow \mathbb{Z}_p$.
Send to the verifier the commitments

$$\begin{aligned} a_d &= \mathrm{com}_{ck}(\boldsymbol{d}_x; r_d) & b_d &= \mathrm{com}_{ck}(\boldsymbol{d}_y; s_d) \\ c_1 &= \mathrm{com}_{ck}(\boldsymbol{x}\boldsymbol{d}_y^\top + \boldsymbol{d}_x\boldsymbol{y}^\top; t_1) & c_0 &= \mathrm{com}_{ck}(\boldsymbol{d}_x\boldsymbol{d}_y^\top; t_0). \end{aligned}$$

P← V: Send challenge $e \leftarrow \mathbb{Z}_p$ to the prover.
P→ V: Send to the verifier the following answer

$$\boldsymbol{f}_x = e\boldsymbol{x}+\boldsymbol{d}_x \quad \boldsymbol{f}_y = e\boldsymbol{y}+\boldsymbol{d}_y \quad r_x = er+r_d \quad s_y = es+s_d \qquad t_z = e^2t+et_1+t_0.$$

V: Accept the argument if

$$a^e a_d = \mathrm{com}_{ck}(\boldsymbol{f}_x; r_x) \wedge b^e b_d = \mathrm{com}_{ck}(\boldsymbol{f}_y; s_y) \wedge c^{e^2} c_1^e c_0 = \mathrm{com}_{ck}(\boldsymbol{f}_x\boldsymbol{f}_y^\top; t_z).$$

Theorem 1. *The protocol above is a 3-move public-coin argument of knowledge of committed values $\boldsymbol{x}, \boldsymbol{y}, z$ so $z = \boldsymbol{x} * \boldsymbol{y}$. The argument has perfect completeness, perfect SHVZK and witness-extended emulation.*

We refer to the full paper for a proof.

4.2 Constant-Round Reduction to the Minimal Case

Next, we give a SHVZK argument that uses a 2-round communication-efficient reduction to the minimal case $m = 1$.

Common input: Commitment key ck and a statement consisting of commitments $a_1, b_1, \ldots, a_m, b_m, c$.

Prover's input: Openings of commitments $\boldsymbol{x}_1, r_1, \boldsymbol{y}_1, s_1, \ldots, \boldsymbol{x}_m, r_m, \boldsymbol{y}_m, s_m, z, t$ so $z = \sum_{i=1}^m \boldsymbol{x}_i * \boldsymbol{y}_i$.

Argument:

P→ V: Prover picks randomizers $t_\ell \leftarrow \mathbb{Z}_p$ for $0 \leq \ell \leq 2m-1$, setting $t_{m-1} = t$ though.

Prover computes $c_0, \ldots, c_{2m-2}$ as

$$c_\ell = \mathrm{com}_{ck}\left(\sum_{i,j\,:\,\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j\ ;\ t_\ell\right).$$

Observe, by construction $c_{m-1} = c$.

Prover sends $c_0, \ldots, c_{2m-2}$ to verifier.

P← V: Verifier sends prover random challenge $e \leftarrow \mathbb{Z}_p$.

P↔ V: Define

$$a' = \prod_{i=1}^m a_i^{e^{i-1}} \qquad b' = \prod_{j=1}^m b_j^{e^{m-j}} \qquad c' = \prod_{\ell=0}^{2m-2} c_\ell^{e^\ell}.$$

Prover computes openings

$$\boldsymbol{x}' = \sum_{i=1}^m e^{i-1}\boldsymbol{x}_i \quad r' = \sum_{i=1}^m e^{i-1} r_i \qquad \boldsymbol{y}' = \sum_{j=1}^m e^{m-j}\boldsymbol{y}_j \quad s' = \sum_{j=1}^m e^{m-j} s_j$$

and

$$z' = \sum_{\ell=0}^{2m-2} e^\ell \sum_{i,j\,:\,\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j \qquad t' = \sum_{\ell=0}^{2m-2} e^\ell t_\ell.$$

Prover and verifier run the minimal case SHVZK argument from Section 4.1 on a', b', c'.

Theorem 2. *The argument above is a public-coin argument for knowledge of openings so $z = \sum_{i=1}^m \boldsymbol{x}_i * \boldsymbol{y}_i$. The argument has perfect completeness, perfect SHVZK and computational witness-extended emulation.*

We refer to the full paper for a proof. Below, we will sketch the main ideas in the construction and why it works.

The important part in the reduction to the minimal case is to use the verifier's challenge in a way such that the prover only needs to send $2m - 2$ commitments to the verifier. We do this by computing a', b' as multi-exponentiations of $a_1, b_1, \ldots, a_m, b_m$ with exponents that are carefully chosen powers of the challenge e. The product of the openings of a' and b' is

$$\left(\sum_{i=1}^{m} e^{i-1}\boldsymbol{x}_i\right) * \left(\sum_{j=1}^{m} e^{m-j}\boldsymbol{y}_j\right) = \sum_{\ell=0}^{2m-2} e^{\ell}\left(\sum_{i,j\,:\,\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j\right).$$

This is the key observation to show that the argument is perfectly complete.

The part corresponding to $\ell = m - 1$ gives us exactly the sum we are after, but we have some extra coefficients of the polynomial corresponding to $\ell \neq m - 1$. To cancel them out, the prover makes $2m - 2$ commitments to these values before seeing the challenge e. Suppose we know openings of all the commitments let us argue that there is negligible probability of correctly answering the challenge e unless indeed $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$. Since all commitments are chosen by the prover before seeing the challenge e, by the binding property of the commitment scheme this shows

$$\sum_{\ell=0}^{2m-2} e^{\ell}\left(\sum_{\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j\right) = \sum_{\ell=0}^{2m-2} e^{\ell} z_{\ell},$$

for random e where $z = z_{m-1}$ since $c = c_{m-1}$. But if $z \neq \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$ the Schwartz-Zippel lemma tells us this can happen with probability at most $\frac{2m-2}{p}$.

EFFICIENCY. The prover sends $2m - 2$ commitments to the verifier. Computing the commitments requires the prover to make $2m - 2$ double-exponentiations and naïvely m^2 bilinear map evaluations to compute the entries to the commitments. Naïvely this requires $m^2 n$ multiplications, but using more advanced techniques such as organizing the vectors in $m \times n$ matrices and using Strassen's matrix multiplication algorithm to compute $XY^\top$ to get the m^2 dot products the cost can be further reduced. However, it is not known how to bring the cost down to $O(mn)$ multiplications.

4.3 Trading Computation for Interaction

Let us again look at the equation

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i.$$

When m is large, the computational overhead of doing the multiplications in the SHVZK argument in the previous section may be prohibitive. In this section, we will trade computational complexity for round complexity by giving a $2\log m$-round reduction to the minimal case that only requires $4mn$ multiplications for the prover.

To illustrate the source of the gain, look at the matrix containing the m^2 products $\boldsymbol{x}_i * \boldsymbol{y}_j$. An example of an 8×8 matrix is given below.

$$\left(\begin{array}{cc|cc|cccc}
\boldsymbol{x}_1 * \boldsymbol{y}_1 & \boldsymbol{x}_1 * \boldsymbol{y}_2 & \boldsymbol{x}_1 * \boldsymbol{y}_3 & \boldsymbol{x}_1 * \boldsymbol{y}_4 & \boldsymbol{x}_1 * \boldsymbol{y}_5 & \boldsymbol{x}_1 * \boldsymbol{y}_6 & \boldsymbol{x}_1 * \boldsymbol{y}_7 & \boldsymbol{x}_1 * \boldsymbol{y}_8 \\
\boldsymbol{x}_2 * \boldsymbol{y}_1 & \boldsymbol{x}_2 * \boldsymbol{y}_2 & \boldsymbol{x}_2 * \boldsymbol{y}_3 & \boldsymbol{x}_2 * \boldsymbol{y}_4 & \boldsymbol{x}_2 * \boldsymbol{y}_5 & \boldsymbol{x}_2 * \boldsymbol{y}_6 & \boldsymbol{x}_2 * \boldsymbol{y}_7 & \boldsymbol{x}_2 * \boldsymbol{y}_8 \\
\boldsymbol{x}_3 * \boldsymbol{y}_1 & \boldsymbol{x}_3 * \boldsymbol{y}_2 & \boldsymbol{x}_3 * \boldsymbol{y}_3 & \boldsymbol{x}_3 * \boldsymbol{y}_4 & \boldsymbol{x}_3 * \boldsymbol{y}_5 & \boldsymbol{x}_3 * \boldsymbol{y}_6 & \boldsymbol{x}_3 * \boldsymbol{y}_7 & \boldsymbol{x}_3 * \boldsymbol{y}_8 \\
\boldsymbol{x}_4 * \boldsymbol{y}_1 & \boldsymbol{x}_4 * \boldsymbol{y}_2 & \boldsymbol{x}_4 * \boldsymbol{y}_3 & \boldsymbol{x}_4 * \boldsymbol{y}_4 & \boldsymbol{x}_4 * \boldsymbol{y}_5 & \boldsymbol{x}_4 * \boldsymbol{y}_6 & \boldsymbol{x}_4 * \boldsymbol{y}_7 & \boldsymbol{x}_4 * \boldsymbol{y}_8 \\
\hline
\boldsymbol{x}_5 * \boldsymbol{y}_1 & \boldsymbol{x}_5 * \boldsymbol{y}_2 & \boldsymbol{x}_5 * \boldsymbol{y}_3 & \boldsymbol{x}_5 * \boldsymbol{y}_4 & \boldsymbol{x}_5 * \boldsymbol{y}_5 & \boldsymbol{x}_5 * \boldsymbol{y}_6 & \boldsymbol{x}_5 * \boldsymbol{y}_7 & \boldsymbol{x}_5 * \boldsymbol{y}_8 \\
\boldsymbol{x}_6 * \boldsymbol{y}_1 & \boldsymbol{x}_6 * \boldsymbol{y}_2 & \boldsymbol{x}_6 * \boldsymbol{y}_3 & \boldsymbol{x}_6 * \boldsymbol{y}_4 & \boldsymbol{x}_6 * \boldsymbol{y}_5 & \boldsymbol{x}_6 * \boldsymbol{y}_6 & \boldsymbol{x}_6 * \boldsymbol{y}_7 & \boldsymbol{x}_6 * \boldsymbol{y}_8 \\
\boldsymbol{x}_7 * \boldsymbol{y}_1 & \boldsymbol{x}_7 * \boldsymbol{y}_2 & \boldsymbol{x}_7 * \boldsymbol{y}_3 & \boldsymbol{x}_7 * \boldsymbol{y}_4 & \boldsymbol{x}_7 * \boldsymbol{y}_5 & \boldsymbol{x}_7 * \boldsymbol{y}_6 & \boldsymbol{x}_7 * \boldsymbol{y}_7 & \boldsymbol{x}_7 * \boldsymbol{y}_8 \\
\boldsymbol{x}_8 * \boldsymbol{y}_1 & \boldsymbol{x}_8 * \boldsymbol{y}_2 & \boldsymbol{x}_8 * \boldsymbol{y}_3 & \boldsymbol{x}_8 * \boldsymbol{y}_4 & \boldsymbol{x}_8 * \boldsymbol{y}_5 & \boldsymbol{x}_8 * \boldsymbol{y}_6 & \boldsymbol{x}_8 * \boldsymbol{y}_7 & \boldsymbol{x}_8 * \boldsymbol{y}_8
\end{array}\right)$$

We want to argue knowledge of c being a commitment to the trace of the matrix. In the SHVZK argument we gave in the previous section, all the $2m-1$ lines that are parallel with the diagonal correspond to entries that have the same degree in the polynomial in e. For instance the sum of the diagonal entries is the coefficient of e^{m-1} whereas the sum of the entries with $i - j = 1$ is the coefficient of e^m. In the SHVZK argument in the previous section, we computed all these m^2 products. Since they each cost n multiplications to compute, we end up using $m^2 n$ multiplications. Even with the best known advanced matrix-multiplication techniques the cost is still significantly higher than $\omega(mn)$ multiplications. As an example, in the 8×8 matrix above we end up computing 64 vector products. We are only interested in the 8 entries along the diagonal, so the remaining computation is just waste that we need to discard in the argument. We will devise a method that allows us to compute larger sub-matrices at once, instead of taking each individual entry at a time. Looking again at the example, if we can discard 2×2 matrices and 4×4 matrices, we only need to discard 14 sub-matrices instead of the 56 entries we need to discard in the reduction in the previous section.

Below, we give a SHVZK argument that reduces the statement to the minimal case $m = 1$ through $\log m$ recursive calls to itself. For simplicity we assume that $m = 2^\mu$. We can do this without loss of generality, because we can always fill up with dummy elements consisting of zero-vectors and trivial commitments, which do not carry any computational overhead.

The idea in the recursive call is to handle the 2×2 matrices along the diagonal at once. We already have a commitment c to the sum of the diagonal entries. In addition, the prover sends commitments c_l, c_u to the verifier, containing respectively the sum of the lower-left corners of the sub-matrices and the sum of the upper-right corners of the sub-matrices along the diagonal.

The verifier responds with a random challenge $e \leftarrow \mathbb{Z}_p$. The prover now reduces her set of vectors to half, by computing

$$\boldsymbol{x}'_i = \boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i} \qquad\qquad \boldsymbol{y}'_i = e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i}.$$

The homomorphic properties of the commitments enables the verifier to compute commitments to these vectors as $a'_i = a_{2i-1}a_{2i}^e$ and $b'_i = b_{2i-1}^e b_{2i}$. We also compute $c' = c_l^{e^2} c^e c_u$, which is a commitment to the sum of the diagonal entries in the new matrix obtained from the vectors $\boldsymbol{x}'_1, \boldsymbol{y}'_1, \ldots, \boldsymbol{x}'_{m/2}, \boldsymbol{y}'_{m/2}$.

The prover and the verifier now engage in a SHVZK argument with these new commitments and vectors for c' containing the sum of the diagonal elements of the matrix. The implication is that for random e we have

$$\sum_{i=1}^{m/2}(\boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i}) * (e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i}) = e^2 z_l + ez + z_u,$$

where z_l and z_u are the contents of c_l and c_u. By the Schwartz-Zippel lemma this implies with overwhelming probability $z = \sum_{i=1}^{m/2}(\boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i-1} + \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i})$, which is what we wanted to prove.

Common input: Commitment key ck and commitments $a_1, b_1, \ldots, a_m, b_m, c$, with $m = 2^\mu$.

Prover's input: Openings of commitments $\boldsymbol{x}_1, r_1, \boldsymbol{y}_1, s_1, \ldots, \boldsymbol{x}_m, r_m, \boldsymbol{y}_m, s_m, z, t$ so $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$.

Argument:

If $m = 1$**:** Run the SHVZK argument from Section 4.1 with common input ck, a_1, b_1, c and prover input $\boldsymbol{x}_1, r_1, \boldsymbol{y}_1, s_1, z, t$ to show $z = \boldsymbol{x}_1 * \boldsymbol{y}_1$.

Else if $m > 1$**:** Define $m' = m/2$ and do

P→ V: Prover picks $t_l, t_u \leftarrow \mathbb{Z}_p$ and sends to verifier

$$c_l = \mathrm{com}_{ck}\Big(\sum_{i=1}^{m'} \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i-1}; t_l\Big) \qquad \text{and} \qquad c_u = \mathrm{com}_{ck}\Big(\sum_{i=1}^{m'} \boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i}; t_u\Big).$$

P← V: Verifier picks random challenge $e \leftarrow \mathbb{Z}_p$ and sends it to prover.

P↔ V: Recursively run argument with common input $ck, a'_1, b'_1, \ldots, a'_{m'}, b'_{m'}, c'$ given by

$$a'_i = a_{2i-1} a_{2i}^e \qquad b'_i = b_{2i-1}^e b_{2i} \qquad c' = c_l^{e^2} c^e c_u.$$

The prover's private input is $\boldsymbol{x}'_1, r'_1, \boldsymbol{y}'_1, s'_1, \ldots, \boldsymbol{x}'_{m'}, r_{m'}, \boldsymbol{y}'_{m'}, s_{m'}, z', t'$ with

$$\boldsymbol{x}'_i = \boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i} \quad r'_i = r_{2i-1} + er_{2i} \quad \boldsymbol{y}'_i = e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i} \quad s'_i = es_{2i-1} + s_{2i}$$

$$z' = e^2 \sum_{i=1}^{m'} \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i-1} + ez + \sum_{i=1}^{m'} \boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i} \qquad t' = e^2 t_l + et + t_u.$$

Theorem 3. *The argument above is a public-coin argument for knowledge of openings so* $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$*. The argument has perfect completeness, perfect SHVZK and computational witness-extended emulation.*

The proof can be found in the full paper.

EFFICIENCY. Each recursive call to the SHVZK argument with $m > 1$ makes the prover send 2 commitments to the verifier. The main computational cost for the prover is the computation of $m = 2m'$ new vectors costing around mn multiplications and m bilinear map evaluations costing around n multiplications each. Summing up over $\log m$

recursive calls, we get a total communication of $2\log m$ commitments from the prover to the verifier and a computational cost for the prover of $4mn$ multiplications in $\mathbb{Z}_p$. The verifier can wait until the proof is over to compute anything; this permits the verifier to use multi-exponentiation techniques for computing the commitments a, b, c that are used in the final call to the minimal case SHVZK argument where $m = 1$. As a consequence, the verifier uses the equivalnt of $4m\kappa/\log m$ multiplications.

5 Zero-Knowledge Arguments for Linear Algebra Equations

We now have several tools to deal with committed matrices and vectors. We can add matrices and vectors using the homomorphic properties of the commitment scheme and we have SHVZK arguments for equations involving multiplications of matrices and vectors and Hadamard products of matrices and vectors. We will sketch how to use these tools to get sub-linear zero-knowledge arguments for equations often arising in linear algebra.

INVERSE. To prove committed matrices satisfy $Y = X^{-1}$ or equivalently $XY = I$, we let the verifier pick $\boldsymbol{s} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and the prover give a SHVZK argument for $(\boldsymbol{s}X)Y = \boldsymbol{s}$.

TRANSPOSE. To prove that a committed matrices satisfy $Y = X^\top$, we let the verifier pick $\boldsymbol{s}, \boldsymbol{t} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and the prover give a SHVZK argument for $(\boldsymbol{s}X)\boldsymbol{t}^\top = (\boldsymbol{t}Y)\boldsymbol{s}^\top$.

EIGENVALUES AND EIGENVECTORS. To show that we have a commitment to an eigenvalue λ and an eigenvector $\boldsymbol{y}^\top$ of X, we first commit to $\boldsymbol{z} = \lambda\boldsymbol{y}$. There are standard SHVZK arguments for $\boldsymbol{z} = \lambda\boldsymbol{y}$, so the prover can show the committed $\boldsymbol{z}$ is correct. Now the verifier picks $\boldsymbol{s} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and we also give a SHVZK argument for $\boldsymbol{s}\boldsymbol{z}^\top = (\boldsymbol{s}X)\boldsymbol{y}^\top$.

SUMS OF ROWS AND COLUMNS. Computing the sum of all row vectors or all column vectors of a matrix corresponds to computing $X\mathbf{1}^\top$ and $\mathbf{1}X$ respectively, where $\mathbf{1} = (1, \ldots, 1)$. The sum of all entries in a matrix can be computed as $\mathbf{1}A\mathbf{1}^\top$. With our techniques we get efficient SHVZK arguments for the correctness of these computations.

HADAMARD PRODUCTS OF ROWS AND COLUMNS. Let us give a SHVZK argument for a committed vector $\boldsymbol{z}$ containing the Hadamard product of all the rows $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ of committed matrix. The prover commits to vectors $\boldsymbol{y}_i = \boldsymbol{x}_1 \circ \cdots \circ \boldsymbol{x}_i$, using $\boldsymbol{y}_1 = \boldsymbol{x}_1$ and $\boldsymbol{y}_n = \boldsymbol{z}$. By demonstrating for $1 \leq i < n$ that $\boldsymbol{y}_{i+1} = \boldsymbol{y}_i \circ \boldsymbol{x}_{i+1}$ we convince the verifier that $\boldsymbol{z}$ is the Hadamard product of the row vectors in X. We remark that it is easy to get a SHVZK argument for $z = \prod_{i=1}^n z_i$, where $\boldsymbol{z} = (z_1, \ldots, z_n)$, so we can extend our SHVZK argument to prove z is the product of all entries in the matrix. In case we want to show $\boldsymbol{z}^\top$ is the Hadamard product of all the columns, we can commit to $X^\top$, using the SHVZK for transposition to prove correctness, and show $\boldsymbol{z}$ is the Hadamard product of all the rows.

TRIANGULARITY. The Hadamard product enables us to prove that a subset of the entries in a committed matrix X consists of all zeroes. Let S be the matrix that has 1 in all entries belong to the subset and has 0 in all other entries. We give a SHVZK argument for $S \circ X = 0$. This SHVZK argument can for instance be used to demonstrate that a committed matrix is lower triangular, upper triangular or diagonal.

TRACE. To show committed values satisfying $z = \text{trace}(X)$ we give a SHVZK argument for $z = \sum_{i=1}^{n} \boldsymbol{s}_i \boldsymbol{x}_i^\top$, where $\boldsymbol{s}_i$ is the ith row vector of I.

ABSOLUTE VALUE OF DETERMINANT. We can commit to an LUP factorization of a matrix X. Proving lower and upper triangularity we already know how to do. It is also easy to prove that we have a committed permutation matrix P, for instance by showing that the matrix is a hidden permutation of I (see Section 5) and that $\mathbf{1}P = \mathbf{1}$ and $P\mathbf{1}^\top = \mathbf{1}^\top$. Since we can single out the diagonal elements of L and U we can compute the determinants of these matrices. We know that P has determinant -1 or 1. We therefore get the determinant up to the sign. We leave it as an open problem to give a sub-linear zero-knowledge argument for the permutation matrix P having determinant -1 or $+1$.

KNOWN PERMUTATION OF A MATRIX. Consider a publicly known permutation π over $\mathbb{Z}_n \times \mathbb{Z}_n$ and two committed matrices $Y = \pi(X)$, meaning for all pairs (i, j) we have $y_{ij} = x_{\pi(ij)}$. To give a SHVZK argument for this, the verifier first picks $R \leftarrow \text{Van}_{n^2}(\mathbb{Z}_p)$ and we ask the prover to show

$$\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} r_{\pi(ij)}y_{ij},$$

which by the Schwartz-Zippel fails with probability $\frac{n^2-1}{p}$ unless indeed $Y = \pi(X)$. Define $S = \pi(R)$ and call the row vectors of the matrices respectively $\boldsymbol{r}_i, \boldsymbol{s}_i, \boldsymbol{x}_i, \boldsymbol{y}_i$. The statement above is equivalent to

$$\sum_{i=1}^{n} \boldsymbol{r}_i \boldsymbol{x}_i^\top = \sum_{i=1}^{n} \boldsymbol{s}_i \boldsymbol{y}_i^\top,$$

for which we already know how to give a SHVZK argument.

HIDDEN PERMUTATION OF A MATRIX. To show that there is a secret permutation π so $Y = \pi(X)$, we use the fact that polynomials are identical under permutation of the roots; an idea that stems from Neff [Nef01]. The verifier picks $r \leftarrow \mathbb{Z}_p$ at random and we let R be the matrix that has r in all entries. We then use the SHVZK argument from Section 5 to show that the product of the entries in $X - R$ equals the product of the entries in $Y - R$. In other words, we show for a random r that

$$\prod_{i=1}^{n}\prod_{j=1}^{n}(x_{ij} - r) = \prod_{i=1}^{n}\prod_{j=1}^{n}(y_{ij} - r),$$

which by the Schwartz-Zippel lemma demonstrates that the two polynomials are identical and thus there exists a permutation π so $x_{ij} = y_{\pi(ij)}$.

This type of SHVZK argument is useful in the context of shuffling [Cha81] and one of the main contributions of Groth and Ishai [GI08] was to show how to give an argument with sub-linear communication. Their SHVZK argument had a communication complexity that could be brought down to $\Theta(n^{4/3})$ group elements at the cost of a super-linear computation complexity of $\Theta(n^{8/3})$ exponentiations.[2] In comparison, our SHVZK argument has a communication complexity of $\Theta(n)$ field elements and even for the constant round protocol we get a much better computation complexity of n^3 *multiplications*. Using a logarithmic number of rounds, we can bring that down to $2n^2\kappa/\log n$ multiplications, beating even the best non-sublinear shuffle argument [Gro03].

6 Circuit Satisfiability

Let us consider an arithmetic circuit built from N addition and multiplication gates over a field $\mathbb{Z}_p$. We want to give a SHVZK argument for the existence of input values to the circuit that makes it evaluate to 1. All gates have two input wires and one output, some of which may be known constants. By introducing dummy gates we can without loss of generality assume $3N = n^2$ and that the number of multiplication gates M and the number of addition gates A are multiples of n.

1. We number the addition gates 1 through A and the multiplication gates $A+1$ through $A+M$. We arrange the inputs and outputs such that the first two rows contain input values to the first n addition gates and the third row contains the corresponding output values, then follows another two rows of input gates and one row of output gates, *etc*. Arranging the circuit in this way, the first $3A$ rows are used for addition gates, while the last $3M$ rows are used for multiplication gates. The prover commits to all these values.
2. For the addition gates, we create the commitment to row $3i$ as the product of the commitments to row $3i-2$ and $3i-1$. By the homomorphic properties of the commitment scheme, this shows that the addition gates are satisfied by the committed wires.
3. For the multiplication gates we can use the SHVZK argument for Hadamard products, to show that the commitment to row $3i$ is the Hadamard product of the commitments to rows $3i-2$ and $3i-1$. This shows that all the multiplication gates are satisfied by the committed values.
4. Some of the values in the matrix may be publicly known constants. By introducing dummy gates and organizing the matrix such that constants appear in the same row, we can without loss of generality assume that we have entire rows that have publicly known constants. We can make these commitments with trivial randomness so the verifier easily can check that the right constants appear in the right places.
5. Finally, we need to demonstrate that all wires appearing many places in the matrix have the same value assigned to them. The output wire of one gate, might for instance appear elsewhere in matrix as an input wire of another gate; we need to give a SHVZK argument for them having the same value. Let us first look at just one wire that appears many places, say coordinates $(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)$. We can create a directed Hamiltonian cycle on this set of indices. Let now π be a permutation that contains directed Hamiltonian cycles for all wires in the circuit.

[2] The computational complexity of Groth and Ishai's shuffle argument can be reduced at the cost of increasing communication.

We use our SHVZK argument for known permutations to show that $X = \pi(X)$. This proves that the committed values are consistent, giving the same value to the same wire everywhere in the matrix.

6.1 Binary Circuits

We have given a SHVZK argument for arithmetic circuit satisfiability, demonstrating the generality of our techniques. The argument consists of committing to a matrix and using some of the SHVZK arguments we have developed in the paper, so it inherits the low communication complexity from the previous sections. The computational complexity of the arithmetic circuit is dominated by the commitment to the wires, costing the prover $O(N\kappa/\log N)$ multiplications.

If we look instead at a binary circuit, where the wires can be 0 or 1, we can reduce the computational complexity. Committing to a binary matrix requires only $O(N/\log N)$ *multiplications* of group elements. Giving a satisfiability argument for a binary circuit requires demonstrating that we have committed to binary values only. This can be done quite easily by demonstrating the committed matrix satisfies $X = X \circ X$.

7 Efficiency

In the following table, we give efficiency estimates for SHVZK arguments we have considered in the paper. We use the parameters κ, κ' and n to represent respectively the size of a field element, the size of a group element and the number of elements in a vector. We assume n is large, since this is where efficient zero-knowledge arguments are most needed and ignore small terms. We measure communication in bits and computation in multiplications in $\mathbb{Z}_p$. We let ρ, ϵ be the costs of respectively a multiplication in G and an addition in $\mathbb{Z}_p$ measured in multiplications in $\mathbb{Z}_p$.

SHVZK argument	Rounds	Communication	Prover computation	Verifier computation
$z = \boldsymbol{x} * \boldsymbol{y}$	3	$2n\kappa$	$4n\frac{\kappa\rho}{\log n}$	$2n\frac{\kappa\rho}{\log n}$
$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$	5	$2n\kappa + 2m\kappa'$	$m^2 n + 4m\frac{\kappa\rho}{\log m} + 4n\frac{\kappa\rho}{\log n}$	$8m\frac{\kappa\rho}{\log m} + 2n\frac{\kappa\rho}{\log n}$
$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$	$2\log m + 3$	$2n\kappa$	$4mn + 4n\frac{\kappa\rho}{\log n}$	$4m\frac{\kappa\rho}{\log m} + 2n\frac{\kappa\rho}{\log n}$
Inverse $Y = X^{-1}$	4	$2n\kappa$	$n^2 + 4n\frac{\kappa\rho}{\log n}$	$4n\frac{\kappa\rho}{\log n}$
Transpose $Y = X^{\top}$	6	$2n\kappa$	$2n^2 + 4n\kappa\rho/\log n$	$6n\frac{\kappa\rho}{\log n}$
Eigenv. $\lambda\boldsymbol{y}^{\top} = X\boldsymbol{y}^{\top}$	5	$5n\kappa$	$n^2 + 12n\frac{\kappa\rho}{\log n}$	$6n\frac{\kappa\rho}{\log n}$
Triangularity	6	$2n\kappa + 2n\kappa'$	$n^3\epsilon + 4n^2 + 8n\frac{\kappa\rho}{\log n}$	$10n\frac{\kappa\rho}{\log n}$
Triangularity	$2\log m + 4$	$2n\kappa$	$6n^2 + 4n\frac{\kappa\rho}{\log n}$	$6n\frac{\kappa\rho}{\log n}$
Trace(X)	5	$2n\kappa + 2n\kappa'$	$n^3\epsilon + 2n^2 + 8n\frac{\kappa\rho}{\log n}$	$10n\frac{\kappa\rho}{\log n}$
Trace(X)	$2\log n + 3$	$2n\kappa$	$4n^2 + 4n\frac{\kappa\rho}{\log n}$	$6n\frac{\kappa\rho}{\log n}$
Hadamard of rows	7	$2n\kappa + 2n\kappa'$	$n^3 + 2n^2\frac{\kappa\rho}{\log n}$	$10n\frac{\kappa\rho}{\log n}$
Hadamard of rows	$2\log n + 5$	$2n\kappa$	$2n^2\frac{\kappa\rho}{\log n}$	$6n\frac{\kappa\rho}{\log n}$
Known $Y = \pi(X)$	6	$2n\kappa + 4n\kappa'$	$4n^3 + 12n\frac{\kappa\rho}{\log n}$	$3n^2 + 14n\frac{\kappa\rho}{\log n}$
Known $Y = \pi(X)$	$2\log n + 4$	$2n\kappa$	$9n^2 + 4n\frac{\kappa\rho}{\log n}$	$3n^2 + 6n\frac{\kappa\rho}{\log n}$
Hidden $Y = \pi(X)$	8	$2n\kappa + 2n\kappa'$	$n^3 + 2n^2\frac{\kappa\rho}{\log n}$	$10n\frac{\kappa\rho}{\log n}$
Hidden $Y = \pi(X)$	$2\log n + 6$	$2n\kappa$	$2n^2\frac{\kappa\rho}{\log n}$	$6n\frac{\kappa\rho}{\log n}$
Arithmetic circuit	7	$O(\sqrt{N}(\kappa + \kappa'))$	$O(N^{3/2} + N\frac{\kappa\rho}{\log N})$	$O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$
Arithmetic circuit	$\log N + 5$	$O(\sqrt{N}\kappa)$	$O(N\frac{\kappa\rho}{\log N})$	$O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$
Binary circuit	7	$O(\sqrt{N}(\kappa + \kappa'))$	$O(N^{3/2}\epsilon + N + \sqrt{N}\frac{\kappa\rho}{\log N})$	$O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$
Binary circuit	$\log N + 5$	$O(\sqrt{N}\kappa)$	$O(N)$	$O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$

Acknowledgment

Yuval Ishai was involved at an early stage of this research and we greatly appreciate the fruitful discussions and his insightful comments.

References

[ALM$^+$98] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3), 501–555 (1998)

[AS98] Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM 45(1), 70–122 (1998)

[BSGH$^+$05] Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Short PCPs verifiable in polylogarithmic time. In: IEEE Conference on Computational Complexity, pp. 120–134 (2005)

[Cha81] Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)

[Din07] Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM 54(3) (2007)

[GI08] Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008), `http://www.daimi.au.dk/~jg/PCPShuffle.pdf`

[GKR08] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: STOC, pp. 113–122 (2008)

[Gro03] Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002), ePrint Archive: `http://eprint.iacr.org/2005/246`

[Gro04] Groth, J.: Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS. Ph.D thesis, pp. xii+119 (2004)

[IKOS07] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC, pp. 21–30 (2007)

[Joh00] Johnson, D.: Challenges for theoretical computer science (2000), `http://www.research.att.com/~dsj/nsflist.html#Crypto`

[Kil92] Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC, pp. 723–732 (1992)

[KR08] Kalai, Y.T., Raz, R.: Interactive pcp. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 536–547. Springer, Heidelberg (2008)

[Lim00] Lim, C.H.: Efficient multi-exponentiation and application to batch verification of digital signatures (2000), `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps`

[Nef01] Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: ACM CCS, pp. 116–125 (2001)

[Ped91] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)

[Pip80] Pippenger, N.: On the evaluation of powers and monomials. SIAM Journal of Computing 9(2), 230–250 (1980)

New Birthday Attacks on Some MACs Based on Block Ciphers*

Zheng Yuan[1,2,**], Wei Wang[3], Keting Jia[3],
Guangwu Xu[4], and Xiaoyun Wang[1,3,***]

[1] Institute for Advanced Study, Tsinghua University, Beijing 100084, China
{zyuan,xiaoyunwang}@mail.tsinghua.edu.cn
[2] Beijing University of Posts and Telecommunications, Beijing 100876, China
[3] Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China
{weiwangsdu,ktjia}@mail.sdu.edu.cn
[4] Department of Electrical Engineering and Computer Science,
University of Wisconsin-Milwaukee, USA
gxu4uwm@uwm.edu

Abstract. This paper develops several new techniques of cryptanalyzing MACs based on block ciphers, and is divided into two parts.

The first part presents new distinguishers of the MAC construction Alred and its specific instance Alpha-MAC based on AES. For the Alred construction, we first describe a general distinguishing attack which leads to a forgery attack directly with the complexity of the birthday attack. A 2-round collision differential path of Alpha-MAC is adopted to construct a new distinguisher with about $2^{65.5}$ chosen messages and $2^{65.5}$ queries. One of the most important results is to use this new distinguisher to recover the internal state, which is an equivalent subkey of Alpha-MAC. Moreover, our distinguisher on Alred construction can be applied to the MACs based on CBC and CFB encryption modes.

The second part describes the first impossible differential attack on MACs-Pelican, MT-MAC-AES and PC-MAC-AES. Using the birthday attack, enough message pairs that produce the inner near-collision with some specific differences are detected, then the impossible differential attack on 4-round AES to the above mentioned MACs is performed. For Pelican, our attack recovers its internal state, which is an equivalent subkey. For MT-MAC-AES, the attack turns out to be a subkey recovery attack directly. The complexity of the two attacks is $2^{85.5}$ chosen messages and $2^{85.5}$ queries. For PC-MAC-AES, we recover its 256-bit key with $2^{85.5}$ chosen messages and 2^{128} queries.

Keywords: MAC, Birthday attack, Distinguishing attack, Forgery attack, Impossible differential cryptanalysis, AES.

* Supported by the National Natural Science Foundation of China (NSFC Grant No. 60525201 and No.90604036) and 973 Project (No.2007CB807902).

** Supported by China Postdoctoral Science Foundation Funded Project (No. 20080430423).

*** To whom correspondence should be addressed.

Part I Distinguishing and Forgery Attacks on Alred and Its AES-Based Instance Alpha-MAC[1]

1 Introduction to Part I

Message Authentication Code (MAC) is a fixed length information used to ensure data integrity and authenticity, and is widely used in network and security protocols, such as IPsec, SNMP, and SSL/TLS. A MAC algorithm takes a secret key and a message of arbitrary length as input, and outputs a short digest. MAC algorithms have been constructed using various approaches, for example, CBC-MAC [11], OMAC [12], TMAC [14], HMAC/NMAC [2], etc.

The MAC construction Alred was introduced by Daemen and Rijmen [8]. Alred is an iterative MAC construction using reduced block ciphers as iteration functions. The secret key, which is used as the key of the block cipher, is applied in the initialization and the finalization, respectively. The internal state is updated by consecutive injections of message blocks. Alpha-MAC is an efficient instance of Alred based on AES [7]. Since AES has been widely used in practice, Alpha-MAC can be easily implemented. For the performance, Alpha-MAC is 2.5 times faster than the popular CBC-MAC with AES.

It was proved that the Alred construction is as strong as the underlying block cipher with respect to key recovery attacks and any forgery attacks not involving inner collisions [8]. Moreover, for Alpha-MAC, any colliding messages of the same size have to be at least 5 blocks long. Recently, Huang et al. exploited the algebraic properties of the AES, constructed internal collisions, and found second preimages for Alpha-MAC, under the assumption that a key or an internal state is known [10]. Biryukov et al. proposed a side-channel collision attack on Alpha-MAC which recovered its internal state, and mounted a selective forgery attack [5].

The main contribution of this part is to present novel distinguishing attacks on the Alred construction and Alpha-MAC, which lead to forgery attacks directly. More importantly, the distinguishing attack on Alpha-MAC can be applied to recover the internal state, and results in a second preimage attack.

There are two kinds of distinguishing attacks on MACs. Preneel and van Oorschot introduced a general distinguishing attack to identify iterated MACs from a random function [17]. Using the birthday paradox, the adversary can detect the internal collision by appending the same one-block message. Another kind of attacks was suggested by Kim et al., which distinguishes the cryptographic primitive embedded in a MAC construction from a random function [13]. Recently, new techniques to identify the underlying hash functions of MACs were proposed [19,20]. For example, distinguishing attacks on HMAC/NMAC-MD5 and MD5-MAC were proposed in [20]. The inner near-collisions are used in the distinguisher which reveals more information than inner collisions. In the same work, they were able to recover partial subkey of the MD5-MAC as well.

[1] By Zheng Yuan, Keting Jia, Wei Wang, and Xiaoyun Wang. See [21] for more details.

Inspired by Wang et al.'s work [19,20], we propose a new idea to detect the inner near-collision with some specific differences, which can be used to identify the cryptographic primitives embedded in MACs. Build upon this idea, two distinguishing attacks on Alred construction and Alpha-MAC are presented in this part. We first describe a distinguishing attack on the Alred construction. This attack is based on the birthday attack [22] which asserts that there exists a collision differential path with some specific differences. This is an inner near-collision which can be recognized with probability 1 by appending another message pair with the same difference. Next, we present a new distinguisher for Alpha-MAC based on a 2-round collision differential path. By combining with the specific differences in the 2-round collision differential path, we then explore a series of tricks to recover the internal state, which is an equivalent subkey. With the recovered subkey, we can obtain the second preimage of Alpha-MAC for any given message M and its MAC value. The complexity of all the attacks of Alpha-MAC is $2^{65.5}$ MAC queries and $2^{65.5}$ chosen messages with a success rate of 0.63. Moreover, the distinguishing attack on the Alred construction can be applied to the MACs based on CBC and CFB encryption modes.

2 Backgrounds and Notations

In this section, we define some notations, and give brief descriptions of the Alred construction and Alpha-MAC.

2.1 Notations

x_i	:	the i-th message word
y_i	:	the state after the i-th iteration
C	:	the output of MAC algorithm
ΔA	:	the XOR difference of A and A'
n	:	the length of the state
l_w	:	the length of the message word
l_m	:	the length of the MAC output
$M\|N$	:	the concatenation of M and N
$\|x\|$	:	the length of a bit string x
$\lceil x \rceil$	:	the smallest integer not less than x
10^j	:	the $(j+1)$-bit sequence $(1\underbrace{00\cdots0}_{j})$

2.2 Alred Construction

The MAC construction Alred [8] is based on a reduced block cipher. The length of the secret key equals to that of the underlying block cipher, and the message length is a multiple of l_w bits.

Given a message $M = (x_1, x_2, \ldots, x_t)$, the Alred construction is as follows.

1. Apply the full block cipher to the state of all-zero block, i. e., $y_0 = \text{Enc}_K(0)$.

2. Perform the following iteration function f for each message word: (a) *Injection Layout*: Map the bits of the message word to an *injection input* that has the same dimensions as a sequence of r-round subkeys of the block cipher. (b) Apply a sequence of r-round block cipher function to the state, and replace the round subkeys with the injection input, i. e., $y_i = f(y_{i-1}, x_i)$, for $i = 1, 2, \ldots, t$.
3. Apply the full block cipher to the state y_t, and truncate the first l_m bits of the state as the output. The final output $C = Trunc(\mathrm{Enc}_K(y_t))$.

2.3 Alpha-MAC Algorithm

Alpha-MAC [8] is a specific instance of the Alred construction with 1-round AES as its iteration function, where $l_w = 32$. Similar to AES, the Alpha-MAC supports key length of 128, 192 or 256 bits.

The message padding method is to append a single bit '1' followed by the minimum bits of '0' such that the length of the result is a multiple of 32. For AES-128, the Injection Layout places the 4 bytes of each message word $x_i = (x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3})$ into a 4×4 array with the form:

$$\begin{pmatrix} x_{i,0} & 0 & x_{i,1} & 0 \\ 0 & 0 & 0 & 0 \\ x_{i,2} & 0 & x_{i,3} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

which acts as the corresponding 128-bit round subkey. The Alpha-MAC round function consists of the four basic transformations of AES in sequence: AddRoundKey (AK), SubBytes (SB), ShiftRows (SR), and MixColumns (MC) [7].

2.4 Related Works

Our work is related to two types of attacks in the literature. They are the general distinguishing-R attack on all iterated MACs proposed by Preneel and van Oorschot [17], and the distinguishing-H attack on HMAC/NMAC-MD5 and MD5-MAC introduced by Wang et al. [20].

Preneel et al. proposed a general forgery attack on iterated MACs by the birthday paradox, which is applicable to all iterated MACs, such as CBC-MAC. Their technique detects all the colliding pairs among $2^{(n+1)/2}$ known text-MAC pairs by the birthday paradox, where n is the bit length of the chaining variable. For each searched collision, i. e., $MAC(K, M) = MAC(K, M')$, a one-block message N is appended to identify whether it is an internal collision by comparing $MAC(K, M||N)$ and $MAC(K, M'||N)$. If an internal collision is found, then a forgery is created since the MACs of $M\|N'$ and $M'\|N'$ are the same. However, this method cannot be used to distinguish the cryptographic primitives embedded in the MAC.

Wang et al. introduced another interesting idea which can distinguish HMAC/NMAC-MD5 without the related-key setting. They also implemented a partial key recovery attack on MD5-MAC. The main strategy of the distinguishing attack is as follows: The adversary first collects enough two-block message pairs $(M\|N, M'\|N)$ to guarantee the appearance of an expected internal near-collision in the first iteration, then detects such a near-collision by changing the second block with enough messages N'. Once the expected inner near-collision is identified, the MAC is known to be based on MD5. The core of the attack is to detect an inner near-collision instead of a collision.

3 Distinguishing and Forgery Attacks on MAC Construction Alred

This section presents distinguishing and forgery attacks on Alred construction. Enlightened by the idea of Wang et al., we can detect a proper output difference as an inner near-collision by the birthday paradox. When the MAC construction is Alred rather than a random function, this kind of inner near-collision can be detected with probability 1 by substituting the last different message pair with another message pair having the same difference. Based on this detected inner near-collision, a forgery attack can be constructed immediately.

3.1 Distinguishing Attack on Alred Construction

The iteration part of Alred construction is based on the r-round block cipher, where the r-round subkeys are substituted by the injection input. The core of our distinguisher is to detect Δy_{j-1}, which is the output difference of $(j-1)$-th iteration. According to the operation between the injection input and the state involved in the iteration function f, the message word difference Δx_j may extinguish Δy_{j-1}, and lead to a collision at the final output. The form of the difference depends on the operation between the injection input and the state; e. g., for Alred based on IDEA or RC6, the operation is modular addition, while for some others, it is XOR. Without loss of generality, we neglect Injection Layout map, and only consider the round number $r = 1$ and the XOR operation between the message word and the state.

As shown in Fig. 1, there is an inner near-collision after round $(j-1)$. When $\Delta x_j = \Delta y_{j-1}$, there will be an internal collision $x_j \oplus y_{j-1} = x'_j \oplus y'_{j-1}$, which can be propagated to the output. If the construction is Alred, we replace the (x_j, x'_j) with a different $(\overline{x_j}, \overline{x'_j})$, where $\Delta \overline{x_j} = \Delta x_j$, the collision still occurs. According to this property, the distinguisher is constructed as follows:

1. Randomly choose a structure $T = \{M^i | M^i = (x^i_1, x^i_2, \ldots, x^i_t)\}$ composed of $2^{(n+1)/2}$ different messages, and query their corresponding MAC values C^i.
2. By the birthday paradox, search a collision $C^a = C^b$, the corresponding messages are M^a and M^b.
3. Counting backwards, suppose that (x^a_j, x^b_j) is the first unmatched pairs of words in (M^a, M^b), i. e., $x^a_j \neq x^b_j$, $M^a = (x^a_1, \ldots, x^a_j, x_{j+1}, \ldots, x_t)$, and $M^b = (x^b_1, \ldots, x^b_j, , x_{j+1}, \ldots, x_t)$. Replace (x^a_j, x^b_j) with another $(\overline{x^a_j}, \overline{x^b_j})$,

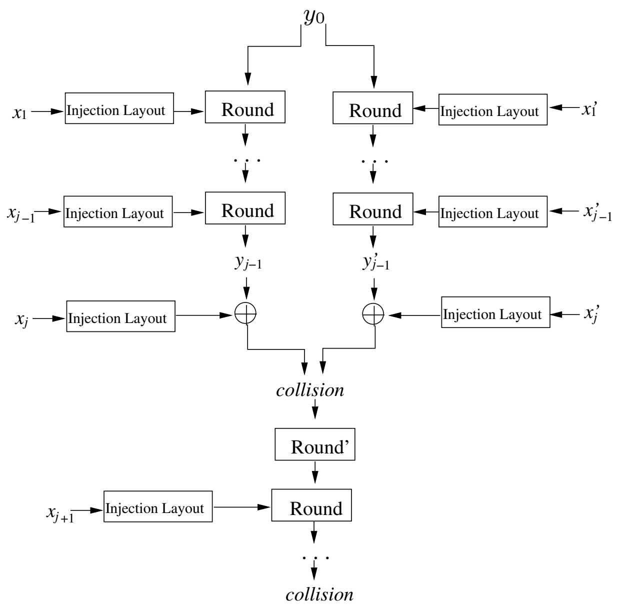


Fig. 1. The Distinguisher with XOR Operation

where $\overline{x_j^a} \oplus \overline{x_j^b} = x_j^a \oplus x_j^b$. Query the MACs with $(\overline{M^a}, \overline{M^b})$, where $\overline{M^a} = (x_1^a, \ldots, x_{j-1}^a, \overline{x_j^a})$ and $\overline{M^b} = (x_1^b, \ldots, x_{j-1}^b, \overline{x_j^b})$.

– If $\overline{C^a} = \overline{C^b}$, we conclude that the MAC is Alred construction.
– Otherwise, it is a random function.

Note that t should be large enough to guarantee the existence of an inner near-collision at round $(j-1)$, where $t \geq 6$.

This attack requires about $2^{(n+1)/2}$ chosen messages, and works with probability 0.63 by the birthday paradox.

Remark 1. For MACs based on the block ciphers with $r \geq 2$, such as CBC-MAC, OMAC, TMAC, etc., the iteration function is $H_i = f(H_{i-1}, x_i) = E_K(H_{i-1} \oplus x_i)$. Therefore, with a little modification, the above attack is applied to these situations. Besides, our method also works for the MACs based on CFB mode, i.e., $H_i = f(H_{i-1}, x_i) = E_K(H_{i-1}) \oplus x_i$.

3.2 Forgery Attack on Alred Construction

Once the inner near-collision is identified, we can replace message words by another pair with the same difference to achieve a new collision pair. Hence, the forgery attack is easily realized with the same complexity and success rate as the distinguishing attack. To be more specific, let (M^a, M^b) be the colliding pair detected in the above distinguishing attack. We query the MAC oracle with $\widetilde{M^a}$, where $\widetilde{M^a} = (x_1^a, \dots, x_{j-1}^a, \widetilde{x_j^a}, s)$, and s is an arbitrary message string. We can get a MAC forgery of the message $\widetilde{M^b} = (x_1^b, \dots, x_{j-1}^b, \widetilde{x_j^a} \oplus \Delta x_j, s)$.

4 Recovering the Equivalent Subkey of Alpha-MAC

It is remarked that the above distinguisher can be utilized to distinguish the Alpha-MAC from a random function. However, we introduce a new distinguisher in this section, where the expected collision implies an inner near-collision with some specific differences. With this distinguisher, we can recover an internal state, which is an equivalent subkey, i. e., the state y_0 (See Fig. 1).

4.1 Some Important Properties of Alpha-MAC

This section introduces a 2-round collision differential path of Alpha-MAC, and summarizes some useful facts based on it. The 2-round differential path will be used to recover the internal state in Section 4.3.

For $i = 1, \dots, t$, denote

$$\begin{pmatrix} y_{i-1,0} & y_{i-1,1} & y_{i-1,2} & y_{i-1,3} \\ y_{i-1,4} & y_{i-1,5} & y_{i-1,6} & y_{i-1,7} \\ y_{i-1,8} & y_{i-1,9} & y_{i-1,10} & y_{i-1,11} \\ y_{i-1,12} & y_{i-1,13} & y_{i-1,14} & y_{i-1,15} \end{pmatrix} \oplus \begin{pmatrix} x_{i,0} & 0 & x_{i,1} & 0 \\ 0 & 0 & 0 & 0 \\ x_{i,2} & 0 & x_{i,3} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{SB}} \begin{pmatrix} z_{i,0} & z_{i,1} & z_{i,2} & z_{i,3} \\ z_{i,4} & z_{i,5} & z_{i,6} & z_{i,7} \\ z_{i,8} & z_{i,9} & z_{i,10} & z_{i,11} \\ z_{i,12} & z_{i,13} & z_{i,14} & z_{i,15} \end{pmatrix},$$

where y_{i-1} is the output of round $(i-1)$, and $(x_{i,0}, 0, x_{i,1}, 0, 0, 0, 0, 0, x_{i,2}, 0, x_{i,3}, 0, 0, 0, 0, 0)$ is the injection input to round i which acts as the round subkeys. Assume that $M=(x_1, x_2, \dots, x_{t-1}, x_t)$ and $M' =(x_1', x_2', \dots, x_{t-1}', x_t')$ follow the 2-round collision differential path as depicted in Fig. 2.

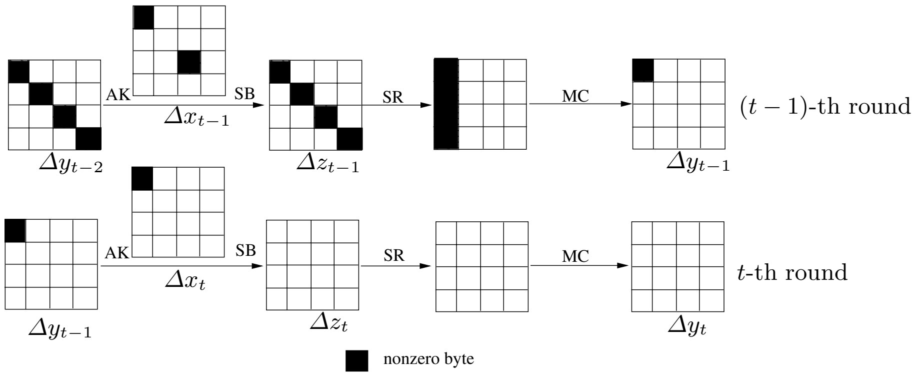


Fig. 2. 2-Round Collision Differential Path

From the differential path, we can see that there is only one nonzero byte in Δy_{t-1}, which equals to $\Delta x_{t,0}$. Because MC is a linear transformation, and SR has no impact on the value of difference, we can compute the output differences of four S-boxes in the $(t-1)$-th round from $\Delta x_{t,0}$:

$$(\Delta z_{t-1,0}, \Delta z_{t-1,5}, \Delta z_{t-1,10}, \Delta z_{t-1,15})^T = MC^{-1}(\Delta x_{t,0}, 0, 0, 0)^T. \qquad (1)$$

Since the branch number of MC transformation in AES is 5 [7], there are four nonzero bytes in Δz_{t-1}, they satisfy the difference structure in Fig. 2. Given the 2-round collision differential path in Fig. 2, we have the following facts:

Fact 1. Given two messages $M{=}(x_1, x_2, \ldots, x_{t-1}, x_t)$ and $M' = (x'_1, x'_2, \ldots, x'_{t-1}, x'_t)$ that follow the 2-round collision differential path, where $\Delta x_t{=}(\Delta x_{t,0}, 0, 0, 0)$, and $\Delta x_{t-1}{=}(\Delta x_{t-1,0}, 0, 0, \Delta x_{t-1,3})$, there exists an *Algorithm* $\boldsymbol{A}_1$ to find another different message pair $\overline{M} = (x_1, x_2, \ldots, \overline{x_{t-1}}, x_t)$ and $\overline{M'} = (x'_1, x'_2, \ldots, \overline{x'_{t-1}}, x'_t)$ satisfying the 2-round collision differential path. Here $(\overline{x_{t-1}}, \overline{x'_{t-1}})$ is obtained by only replacing $(x_{t-1,0}, x'_{t-1,0})$ with different $(\overline{x_{t-1,0}}, \overline{x'_{t-1,0}})$. The complexity of the algorithm is about 2^9 queries and 2^9 chosen messages.

Proof. Since only $(x_{t-1,0}, x'_{t-1,0})$ changes, all bytes in Δz_{t-1} remain the same except $\Delta z_{t-1,0}$, where $\Delta z_{t-1,0} = S(y_{t-2,0} \oplus x_{t-1,0}) \oplus S(y'_{t-2,0} \oplus x'_{t-1,0})$. Thus, M and M' collide if and only if $S(y_{t-2,0} \oplus \overline{x_{t-1,0}}) \oplus S(y'_{t-2,0} \oplus \overline{x'_{t-1,0}}) = \Delta z_{t-1,0}$. From the distribution table of the S-box in AES, we observe that, there are 2^7 pairs corresponding to each output difference on average. Hence, each randomly chosen pair $(x_{t-1,0}, x'_{t-1,0})$ leads to the expected output difference $\Delta z_{t-1,0}$ with probability $2^7/2^{15} = 2^{-8}$. So *Algorithm* $\boldsymbol{A}_1$ needs about 2^8 chosen message pairs $(\overline{M}, \overline{M'})$ and 2^9 corresponding MACs to find the message pair $(\overline{M}, \overline{M'})$ which follows the 2-round collision differential path. □

Fact 1 will be used in the new distinguishing attack to identify the ALPHA-MAC from a random function. And the following Fact can recover two bytes of the unknown internal state y_{t-2} corresponding to the nonzero bytes of Δx_{t-1}.

Fact 2. Given two messages $M{=}(x_1, x_2, \ldots, x_{t-1}, x_t)$ and $M' = (x'_1, x'_2, \ldots, x'_{t-1}, x'_t)$ that follow the 2-round collision differential path, where $\Delta x_t{=}(\Delta x_{t,0}, 0, 0, 0)$, and $\Delta x_{t-1}{=}(\Delta x_{t-1,0}, 0, 0, \Delta x_{t-1,3})$, there exists an *Algorithm* $\boldsymbol{A}_2$ to recover $(y_{t-2,0}, y'_{t-2,0})$ with about 2^{16} XOR operations and 2^9 chosen messages.

Proof. Algorithm $\boldsymbol{A}_2$ is described as follows.

1. Call *Algorithm* $\boldsymbol{A}_1$ to find another message pair $\overline{M} = (x_1, x_2, \ldots, \overline{x_{t-1}}, x_t)$ and $\overline{M'} = (x'_1, x'_2, \ldots, \overline{x'_{t-1}}, x'_t)$ which produce a 2-round collision differential path in Fig. 3. Here $(\overline{x_{t-1}}, \overline{x'_{t-1}})$ are only different at byte position 0 from (x_{t-1}, x'_{t-1}).
2. Compute $z_{t-1,0}$ from Eq. (1), guess all 2^{16} possibilities of $(y_{t-2,0}, y'_{t-2,0})$, and check if the following two equations hold.

$$S(y_{t-2,0} \oplus x_{t-1,0}) \oplus S(y'_{t-2,0} \oplus x'_{t-1,0}) = \Delta z_{t-1,0}, \qquad (2)$$

$$S(y_{t-2,0} \oplus \overline{x_{t-1,0}}) \oplus S(y'_{t-2,0} \oplus \overline{x'_{t-1,0}}) = \Delta z_{t-1,0}. \qquad (3)$$

3. If there is only one solution $(y_{t-2,0}, y'_{t-2,0})$ satisfying Eq. (2) and (3) among 2^{16} guesses, outputs $(y_{t-2,0}, y'_{t-2,0})$.
   Otherwise, repeat steps 1 and 2 until only one solution is left.

It is obvious that, the time complexity of *Algorithm* $\boldsymbol{A}_2$ is about 2^{16} XOR operations and 2^9 chosen messages. □

Fact 3. Given two messages $M=(x_1, x_2, \ldots, x_{t-1}, x_t)$ and $M' = (x'_1, x'_2, \ldots, x'_{t-1}, x'_t)$ that follow the 2-round collision differential path, where $\Delta x_t=(\Delta x_{t,0}, 0, 0, 0)$, and $\Delta x_{t-1}=(\Delta x_{t-1,0}, 0, 0, \Delta x_{t-1,3})$, there exists an *Algorithm* $\boldsymbol{A}_3$ to recover $(y_{t-2,10}, y'_{t-2,10})$ with about 2^{16} XOR operations and 2^9 queries.

Proof. The proof of Fact 3 is similar to that of Fact 2. We only need to replace $(x_{t-1,10}, x'_{t-1,10})$ by different $(\overline{x_{t-1,10}}, \overline{x'_{t-1,10}})$. □

4.2 Distinguishing Attack on Alpha-MAC

Similar to the distinguisher for Alred construction, the new distinguisher on Alpha-MAC is based on the identification of an inner near-collision Δy_{t-1} as shown in Fig. 2. By the birthday paradox, such an inner near-collision exists, and can be detected by the new distinguisher. From *Algorithms* $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$, we can recover two bytes of the internal state y_{t-2}. Moreover, we explore a series of tricks to recover more bytes of the internal state y_{t-3}, and further recover y_0. It is noted that $y_0 = Enc_K(0)$ equals to a subkey used in the secret prefix method.

It is claimed that an extinguishing differential in Alpha-MAC spans at least 5 message words, and given the state value y_{i-1}, the map from the sequence of four message words $(x_i, x_{i+1}, x_{i+2}, x_{i+3})$ to the state value before the $(i+4)$-th iteration is a bijection [8]. Hence, we choose a structure composed of $2^{64.5}$ messages with t-word length, where t is required to be bigger than or equal to 6 in order to guarantee the map from $(x_1, \ldots, x_{t-1})$ to y_{t-1} is a random function, and to ensure the existence of an inner near-collision. It is recommended to choose $t = 9$.

Given a fixed word difference $(\eta, 0, 0, 0)$, construct two structures as follows:

$$T_1 = \{M^a = (x^a_1, x^a_2, \ldots, x^a_{t-1}, x_t)\},$$
$$T_2 = \{M^b = (x^b_1, x^b_2, \ldots, x^b_{t-1}, x_t \oplus (\eta, 0, 0, 0))\},$$

where the message words (x^a_i, x^b_i) $(i = 1, 2, \ldots, t-2)$, $(x^a_{t-1,0}, x^a_{t-1,3})$ of x^a_{t-1}, and $(x^b_{t-1,0}, x^b_{t-1,3})$ of x^b_{t-1} are randomly chosen, and other bytes of (x^a_{t-1}, x^b_{t-1}) are fixed, i. e., we choose Δx_{t-1} and Δx_t as shown in Fig. 2. The distinguisher works in the following manner:

1. Query the MAC with all the $2^{65.5}$ messages in structure T_1 and T_2, and obtain the corresponding MACs.
2. Search for (M^a, M^b) such that $C^a = C^b$ by the birthday attack, where $M^a \in T_1$, $M^b \in T_2$. Randomly choose another different pair $(\overline{M^a}, \overline{M^b})$, where $\overline{M^a} = (x^a_1, \ldots, x^a_{t-1}, \overline{x^a_t})$, $\overline{M^b} = (x^b_1, \ldots, x^b_{t-1}, \overline{x^b_t})$, $\Delta\overline{x_t} = \Delta x_t$. Query

the MAC with the new message pair $(\overline{M^a}, \overline{M^b})$. If $(\overline{M^a}, \overline{M^b})$ is a collision, we conclude that the MAC is Alred-MAC, and go to step 3. Otherwise, the MAC is a random function.

3. Randomly choose 2^8 different $(\overline{x^a_{t-1,0}}, \overline{x^b_{t-1,0}})$ to replace $(x^a_{t-1,0}, x^b_{t-1,0})$. Query the MACs of the new messages. Check whether there is at least one collision among them. If a collision appears, the Alred construction is claimed as the Alpha-MAC. Otherwise, it is other Alred MAC instance.

Complexity Evaluation. Step 1 takes $2^{65.5}$ MAC queries. There are only 2 queries and $2^{65.5}$ table look-ups with $2^{65.5}$ entries in step 2, and 2^8 MAC queries in step 3. Thus, the total complexity is dominated by step 1, which is about $2^{65.5}$ MAC queries and $2^{65.5}$ chosen messages.

Success Rate. A collision between the two structures occurs with probability 0.63 from the birthday paradox, which is also the success rate of our attack.

4.3 Internal State Recovery of Alpha-MAC

In this section, we recover the internal state y_0 combining the new distinguisher discussed above with some new tricks. Once the Alred construction is identified as the Alpha-MAC, we obtain a message pair (M^a, M^b), which follows the 2-round collision differential path (See Fig. 2).

Denote $M^a = (x^a_1, x^a_2, \ldots, x^a_{t-1}, x^a_t)$ and $M^b = (x^b_1, x^b_2, \ldots, x^b_{t-1}, x^b_t)$. The process of the internal state recovery attack is depicted in Fig. 3, where '$*$' denotes the difference that can be computed, '?' stands for the unknown difference, and '0' means zero difference. The details of the recovery attack are as follows:

1. **Recovering** $(y^a_{t-2,0}, y^b_{t-2,0}, y^a_{t-2,10}, y^b_{t-2,10})$.
   By Algorithms $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$, the corresponding bytes $(y^a_{t-2,0}, y^b_{t-2,0}, y^a_{t-2,10}, y^b_{t-2,10})$ can be recovered directly.
   Next, let us explore more techniques to recover more bytes of the internal states y_{t-2} and y_{t-3}.

$$\Delta y_{t-3} = \begin{pmatrix} * & ? & * & ? \\ ? & * & ? & * \\ * & ? & * & ? \\ ? & * & ? & * \end{pmatrix}$$

$$\xleftarrow{AK^{-1}\ SB^{-1}} \Delta z_{t-2} = \begin{pmatrix} * & ? & * & ? \\ ? & * & ? & * \\ * & ? & * & ? \\ ? & * & ? & * \end{pmatrix} \xleftarrow{SR^{-1}\ MC^{-1}} \Delta y_{t-2} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & ? & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & ? \end{pmatrix}$$

$$\xleftarrow{AK^{-1}\ SB^{-1}} \Delta z_{t-1} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{pmatrix} \xleftarrow{SR^{-1}\ MC^{-1}} \Delta y_{t-1} = \begin{pmatrix} \Delta x_{t,0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 3. Recovering the Internal State

2. **Recovering** $(y^a_{t-3,0}, y^b_{t-3,0}, y^a_{t-3,2}, y^b_{t-3,2}, y^a_{t-3,8}, y^b_{t-3,8}, y^a_{t-3,10}, y^b_{t-3,10})$.
Applying MC^{-1} and SR^{-1} to the state y_{t-2}, we obtain the values $(\Delta z_{t-2,0}, \Delta z_{t-2,5}, \Delta z_{t-2,10}, \Delta z_{t-2,15}, \Delta z_{t-2,2}, \Delta z_{t-2,7}, \Delta z_{t-2,8}, \Delta z_{t-2,13})$ from the following two equations:

$$(\Delta z_{t-2,0}, \Delta z_{t-2,5}, \Delta z_{t-2,10}, \Delta z_{t-2,15})^T = MC^{-1}(\Delta y_{t-2,0}, 0, 0, 0)^T, \quad (4)$$
$$(\Delta z_{t-2,2}, \Delta z_{t-2,7}, \Delta z_{t-2,8}, \Delta z_{t-2,13})^T = MC^{-1}(0, 0, \Delta y_{t-2,10}, 0)^T. \quad (5)$$

(a) Recovering $(y^a_{t-3,0}, y^b_{t-3,0})$.
   i. Look up the differential distribution table of AES S-box, and obtain about 2^8 possible values $(y_{t-3,0}, y'_{t-3,0})$ satisfying

$$S(y_{t-3,0} \oplus x^a_{t-2,0}) \oplus S(y'_{t-3,0} \oplus x^b_{t-2,0}) = \Delta z_{t-2,0}, \quad (6)$$

   The correct $(y^a_{t-3,0}, y^b_{t-3,0})$ must be among the 2^8 possible $(y_{t-3,0}, y'_{t-3,0})$.
   ii. Detect the correct $(y^a_{t-3,0}, y^b_{t-3,0})$ from the following fact.

Fact 4. For each possible $(y_{t-3,0}, y'_{t-3,0})$, set $x_{t-2}=(x_{t-2,0}, x^a_{t-2,1}, x^a_{t-2,2}, x^a_{t-2,3})$ and $x'_{t-2} = (x'_{t-2,0}, x^b_{t-2,1}, x^b_{t-2,2}, x^b_{t-2,3})$, where

$$x_{t-2,0} = x^b_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0},$$
$$x'_{t-2,0} = x^a_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0}.$$

Suppose $\overline{\Delta x_{t-1}} = x_{t-1} \oplus x'_{t-1}$ and $\Delta x_{t-1} = x^a_{t-1} \oplus x^b_{t-1}$.
Select 2^8 different word pairs (x_{t-1}, x'_{t-1}) such that

$$\overline{\Delta x_{t-1,0}} \neq \Delta x_{t-1,0} \text{ and } \overline{\Delta x_{t-1,i}} = \Delta x_{t-1,i} \ (i = 1, 2, 3).$$

Query the MAC with 2^8 message pairs (M, M'), where

$$M=\{x^a_1, \ldots, x^a_{t-3}, x_{t-2}, x_{t-1}, x^a_t\} \text{ and } M'=\{x^b_1, \ldots, x^b_{t-3}, x'_{t-2}, x'_{t-1}, x^b_t\}.$$

– If one collision is found among 2^8 message pairs (M, M'), the correct $(y^a_{t-3,0}, y^b_{t-3,0}) = (y_{t-3,0}, y'_{t-3,0})$.
– Otherwise, the $(y_{t-3,0}, y'_{t-3,0})$ is not correct.

Proof. If $(y_{t-3,0}, y'_{t-3,0}) = (y^a_{t-3,0}, y^b_{t-3,0})$, the two inputs to S-boxes are

$$x_{t-2,0} \oplus y^a_{t-3,0} = x^b_{t-2,0} \oplus y^a_{t-3,0} \oplus y^b_{t-3,0} \oplus y^a_{t-3,0} = x^b_{t-2,0} \oplus y^b_{t-3,0},$$
$$x'_{t-2,0} \oplus y^b_{t-3,0} = x^a_{t-2,0} \oplus y^a_{t-3,0} \oplus y^b_{t-3,0} \oplus y^b_{t-3,0} = x^a_{t-2,0} \oplus y^a_{t-3,0},$$

respectively, and the corresponding outputs are

$$S(x_{t-2,0} \oplus y_{t-3,0}) = z_{t-2,0} = z^b_{t-2,0}, \ S(x'_{t-2,0} \oplus y'_{t-3,0}) = z'_{t-2,0} = z^a_{t-2,0},$$

i. e., $\overline{\Delta z_{t-2,0}} = \Delta z_{t-2,0}$, which implies that $\overline{\Delta z_{t-2}} = \Delta z_{t-2}$.

It is noted that, the byte $z_{t-2,0}$ only affects four bytes $(y_{t-2,0}, y_{t-2,4}, y_{t-2,8}, y_{t-2,12})$, which means that the 2nd to 4th columns of $\overline{\Delta y_{t-1}}$ are the same as Δy_{t-1}. Therefore, $y_t = y'_t$ if and only if

$$S(y_{t-2,0} \oplus x_{t-1,0}) \oplus S(y'_{t-2,0} \oplus x'_{t-1,0}) = \Delta z_{t-1,0}. \tag{7}$$

There exists one collision among 2^8 different pair (M, M') on average. If $(y_{t-3,0}, y'_{t-3,0}) \neq (y^a_{t-3,0}, y^b_{t-3,0})$, the two inputs to the S-box are

$$x_{t-2,0} \oplus y^a_{t-3,0} = x^b_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0} \oplus y^a_{t-3,0},$$
$$x'_{t-2,0} \oplus y^b_{t-3,0} = x^a_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0} \oplus y^b_{t-3,0}.$$

The equation $\overline{\Delta z_{t-2,0}} = \Delta z_{t-2,0}$ holds with probability 2^{-8}.

Thus, to guarantee a collision occur, it is required that (i) $\overline{\Delta y_{t-2,4}} = 0$, $\overline{\Delta y_{t-2,8}} = 0$ and $\overline{\Delta y_{t-2,12}} = 0$ when $\overline{\Delta z_{t-2}} \neq \Delta z_{t-2}$, or (ii) Eq. (6) holds when $\overline{\Delta z_{t-2}} = \Delta z_{t-2}$. Among 2^8 different message pairs (M, M'), there is a collision with probability $2^{-24} \times 2^8 = 2^{-16}$ for the first case, and the probability is at most 2^{-8} for the second. □

(b) In a similar manner, the values $(y^a_{t-3,2}, y^b_{t-3,2})$, $(y^a_{t-3,8}, y^b_{t-3,8})$ and $(y^a_{t-3,10}, y^b_{t-3,10})$ can be each filtered by 2^8 message pairs.

3. **Recovering** $(y^a_{t-3,5}, y^b_{t-3,5}, y^a_{t-3,7}, y^b_{t-3,7}, y^a_{t-3,13}, y^b_{t-3,13}, y^a_{t-3,15}, y^b_{t-3,15})$**.** Compute the correct $(y^a_{t-3,5}, y^a_{t-3,15}, y^b_{t-3,5}, y^a_{t-3,15})$ by

$$\begin{aligned}
\Delta z_{t-2,5} &= S(y^a_{t-3,5}) \oplus S(y^b_{t-3,5}),\\
\Delta z_{t-2,15} &= S(y^a_{t-3,15}) \oplus S(y^b_{t-3,15}),\\
y^a_{t-2,0} &= 3S(y^a_{t-3,0} \oplus x^a_{t-2,0}) \oplus 2S(y^a_{t-3,5}) \oplus S(y^a_{t-3,10} \oplus x^a_{t-2,3}) \oplus S(y^a_{t-3,15}),\\
y^b_{t-2,0} &= 3S(y^b_{t-3,0} \oplus x^b_{t-2,0}) \oplus 2S(y^b_{t-3,5}) \oplus S(y^b_{t-3,10} \oplus x^b_{t-2,3}) \oplus S(y^b_{t-3,15}).
\end{aligned}$$

Similarly, obtain the correct $(y^a_{t-3,7}, y^a_{t-3,13}, y^b_{t-3,7}, y^a_{t-3,13})$ from

$$\begin{aligned}
\Delta z_{t-2,7} &= S(y^a_{t-3,7}) \oplus S(y^b_{t-3,7}),\\
\Delta z_{t-2,13} &= S(y^a_{t-3,13}) \oplus S(y^b_{t-3,13}),\\
y^a_{t-2,10} &= S(y^a_{t-3,2} \oplus x^a_{t-2,1}) \oplus S(y^a_{t-3,7}) \oplus 3S(y^a_{t-3,8} \oplus x^a_{t-2,2}) \oplus 2S(y^a_{t-3,13}),\\
y^b_{t-2,10} &= S(y^b_{t-3,2} \oplus x^b_{t-2,1}) \oplus S(y^b_{t-3,7}) \oplus 3S(y^b_{t-3,8} \oplus x^b_{t-2,2}) \oplus 2S(y^b_{t-3,13}).
\end{aligned}$$

4. **Recovering the internal state** y_0**.**
Guess all the 2^{64} possibilities of the rest 8 bytes of y^a_{t-3}. Take $(x^a_{t-3}, \ldots, x^a_1)$ as the decryption subkey, and obtain 2^{64} y_0. For each y_0, compute the corresponding y^b_{t-3} with $(x^b_1, \ldots, x^b_{t-3})$ to filter out the wrong guesses.
If there are more than one y_0 left, using the distinguisher to get another colliding pair, and repeat the whole recovery attack until there is only one value left. Two colliding pairs are enough to sieve the right y_0.

Until now, the recovery attack on the internal state y_0 is completed.

Complexity Evaluation. The complexity of this attack is dominated by the distinguishing attack and the final exhaustive search, which is about $2^{65.5}$ queries and $2^{65.5}$ chosen messages.
Second Preimages for Alpha-MAC. Once the internal state y_0 is recovered, the second preimages can be found by Huang et al.'s attack [10], and a selective forgery attack can be performed as in [5].

5 Conclusion

In this part, the distinguishing and forgery attacks on the Alred construction and its specific instance Alpha-MAC are presented. The complexity of the attacks is dominated by the birthday attack, far less than the exhaustive search. Our contribution is to detect inner near-collisions with specific differences rather than collisions, from which more information can be obtained. Especially for Alpha-MAC, combining with the distinguishing attack, we explore a series of tricks to recover the internal state, which equals to an equivalent subkey. This leads to the second preimage attack on Alpha-MAC. It is remarked that the distinguishing and forgery attacks on Alred construction are also applicable to the MACs based on CBC and CFB encryption modes.

Part II Impossible Differential Cryptanalysis of Pelican, MT-MAC-AES and PC-MAC-AES[2]

6 Introduction to Part II

Besides the MACs introduced in Part I, there are several others based on reduced block ciphers, such as Pelican [9], MT-MAC-AES and PC-MAC-AES [15], and all of them take the 4-round AES [3] as the iteration function.

Pelican is an optimized version of Alpha-MAC, which was proposed by Daemen and Rijmen. It generates the MAC value in a CBC-like manner. The side-channel collision attack on Alpha-MAC works for Pelican, too [5]. Minematsu and Tsunoo also proposed two provably secure MAC constructions, MT-MAC and PC-MAC, which make use of the provably secure almost universal hash functions (AU_2). The MT-MAC uses differentially uniform permutations such as 4-round AES with independent keys in a Wegman-Carter binary tree. However, it is not memory efficient. A modified version PC-MAC, which is based on a CBC-like AU_2 hash PCH (Periodic CBC Hash), was suggested.

Inspired by recent MAC cryptanalysis techniques of Wang et al. [19,20] and the methods introduced in Part I [21], we observe that the impossible differential attack can be extended to MACs provided that enough inner near-collisions with specific differences are detected.

[2] By Wei Wang, Xiaoyun Wang, and Guangwu Xu. See [18] for the full version.

[3] The MT-MAC-AES and PC-MAC-AES take the *simplified* 4-round AES described in Section 7.2.

Impossible differential attack [3] is one of the widely used cryptanalytic techniques on block ciphers. It is a sieving attack which focuses on a differential path with probability 0. If a pair of messages is encrypted or decrypted to an impossible difference under some trial key, one can filter out this trial key from the key space. Thus, the correct key is found by eliminating all the wrong keys which lead to a contradiction. For MAC algorithms, the secret key is usually replaced by the internal state. It seems that, the impossible differential attack is hard to work with MAC algorithms, due to the fact that the internal state values as well as their differences, are concealed by the final full encryption or complex keyed iterations. However, the recent techniques based on the birthday attack overcome this obstacle. One can recognize the inner near-collisions with some specific differences, hence the impossible differential attack can be performed with the detected inner near-collisions.

Taking 4-round AES as a building block, we are able to recover its secret state utilizing a 3-round impossible differential characteristic. For Pelican, the secret subkey is replaced by the internal state, thus we can recover its internal state with $2^{85.5}$ chosen messages and $2^{85.5}$ queries. This attack can be further extended to a subkey recovery attack on MT-MAC-AES with the same complexities. For PC-MAC-AES, we recover its two secret keys separately once the internal state is sieved, with $2^{85.5}$ chosen messages and 2^{128} queries. We emphasis that our results do not contradict to any security proof associated with the designs. Due to space limitations, we only present attacks on Pelican and PC-MAC-AES, while the attack on MT-MAC-AES appears in [18].

7 Backgrounds

Beside the notations defined in Part I, we will use the following notations in this part: let z_i^I denote the input of the i-th AES round, while z_i^B, z_i^R, z_i^M and z_i^O denote the intermediate values after the application of SB, SR, MC and AK of the i-th AES round, respectively. z_i is exhibited as a 4×4 two dimensional array of bytes indexed as:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Next, we give brief descriptions of Pelican and PC-MAC-AES.

7.1 Pelican Algorithm

Pelican is a specific instance of Alred construction taking 4-round AES as a building block [9]. It supports 128-bit block and 128/160/192/224/256-bit key. Pelican takes a message of arbitrary length as input, and outputs a MAC value with length up to 128-bit.

To construct the MAC, let us pad a message M of any length to a multiple of 128-bit by appending a single bit '1' followed by the minimum bits of '0', and split the padded message into 128-bit words $(x_1, x_2, \ldots, x_b)$. The Pelican MAC function works as follows:

1. **Initialization:** Fill the 128-bit *state* with zeros, and encrypt the zero state with AES encryption, i. e., $y_0 = E_K(0)$, where E is the AES, and K is the secret key.
2. **Chaining:** XOR the first message word x_1 to the state, i. e., $y_1 = y_0 \oplus x_1$. For each message word x_i $(i = 2, \ldots, b)$, perform an iteration operation: $y_i = f(y_{i-1}) \oplus x_i$, where f consists of 4-round AES with the round subkeys set to 0.
3. **Finalization:** Apply the full AES to the state, and take the first l_m bits of the state as the MAC value of M. The final output C is $C = Trunc(E_K(y_m))$.

7.2 PC-MAC-AES

PC-MAC is a provably secure MAC construction proposed in [15]. It is composed of an n-bit block cipher E_K, and an n-bit auxiliary keyed permutation G_U. Two secret keys are required, one for the block cipher and the other for making the block cipher tweakable.

For $i = 1, 2, \ldots, s$, let F_i be an n-bit random function. Suppose $x = (x_1, x_2, \ldots, x_{s+1})$, we first define the chaining function:

$$Ch[F_1, \ldots, F_s](x) = x_{s+1} \oplus F_s(x_s \oplus F_{s-1}(\cdots F_2(x_2 \oplus F_1(x_1)) \cdots)),$$

which is used iteratively when the input is longer than $(s+1)$ blocks, and terminates as soon as the last input block is XORed. The PCH is defined as follows:

Definition 1 (Periodic CBC Hash (PCH) [15]).
Let E_K be an n-bit block cipher. For $d \geq 0$, let $G = (G_1, \ldots, G_d)$ be the sequence of keyed auxiliary permutation, where for G_i $(i = 1, \ldots, d)$, the subkey involved in G is U_i. We assume that $(K_1^{XOR}, \ldots, K_{d-1}^{XOR})$ are (d-1) n-bit subkeys. The Periodic CBC Hash is defined as:

$$PCH_d[E_K, G] = Ch[E_K, G_1, G_2^{\oplus K_1^{XOR}}, \ldots, G_d^{\oplus K_{d-1}^{XOR}}].$$

Here, $G_i^{\oplus K_{i-1}^{XOR}}(\alpha) = G_i(\alpha \oplus K_{i-1}^{XOR})$ $(i = 2, \ldots, d)$, where α is an n-bit variable. $PCH_d[E_K, G]$ terminates as soon as the last input block is XORed.

The next is the description of the PC-MAC$_d[E_K|G_U]$ construction.

- **Preprocessing:**
  - Compute $U = (U_1, \ldots, U_d)$, which is the first dl bits of $E_K(0 \oplus L), \ldots, E_K(\hat{a} \oplus L)$. Here, K, L are the secret keys, and $\hat{a} = \lceil dl/n \rceil$.
  - Compute $K_{j-\hat{a}+1}^{\text{XOR}} = E_K(j \oplus L)$, for $j = \hat{a}, \ldots, \hat{a} + d - 2$.

– **MAC Computation:** For a message M with arbitrary length,

$$C = \begin{cases} E_K(PCH_d[E_K, G](M) \oplus L \cdot u) & if\ |M| \mod n = 0, \\ E_K(PCH_d[E_K, G](M \| 10^t) \oplus L \cdot u^2) & if\ |M| \mod n = n - t - 1, \end{cases}$$

where u is an element of $GF(2^n)$ that is not 0 or 1, and $L \cdot u$ is the multiplication of L and u on $GF(2^n)$.

The authors of [15] recommended to implement block cipher E_K with the AES-128, and the permutation G_U with the *simplified* 4-round AES, where the transformations of each round perform in the order of AK, SB, SR and MC, and the addition of the first round key and the last diffusion layer are omitted. We call this AES-based instance PC-MAC-AES.

8 Main Idea of the Impossible Differential Cryptanalysis

Similar to the cryptanalysis of block ciphers, to implement an impossible differential attack on MACs, we need to find an impossible differential path first. Then collect many structures of chosen messages, query MACs with them, and sieve the message pairs with the required intermediate differences. For each sieved pair, discard the wrong subkeys (or internal states) which cause the partial encryption and decryption to match the impossible differential path. Finally, after enough pairs are analyzed, only the correct subkey (or internal state) is left.

8.1 Three-Round Impossible Differential Property of AES

For AES, several 4-round impossible differential paths have been found in literature, e. g. [1,4,16]. However, we note that, among the MAC algorithms presented in the previous section, the 4-round AES is taken as a building block. Thus, we focus on the reduced AES and only need a 3-round impossible differential path.

The 3-round impossible differential path is stated as follows.

Property 1 (Impossible Differential Path of 3-round AES). For 3-round AES, given an input pair $(z_2^I, z_2^{I\prime})$ whose components equal in all except six bytes indexed by $(0, 1, 5, 8, 12, 13)$ (or $(0, 1, 4, 5, 9, 12)$, $(0, 4, 5, 8, 9, 13)$, $(1, 4, 8, 9, 12, 13)$), the difference of the output pair $(z_4^O, z_4^{O\prime})$ can not have exactly one nonzero byte.

The correctness of Property 1 can be easily proved, and Fig. 4 illustrates the impossible differential path for the case of $(0, 1, 5, 8, 12, 13)$.

8.2 Message Pairs Collection Phase

In the cryptanalysis of block ciphers, we can collect the message pairs available to the impossible differential attack directly according to the output differences and chosen message differences. While for MACs mentioned above, we have to explore new techniques to collect such message pairs since the 4-round AES is

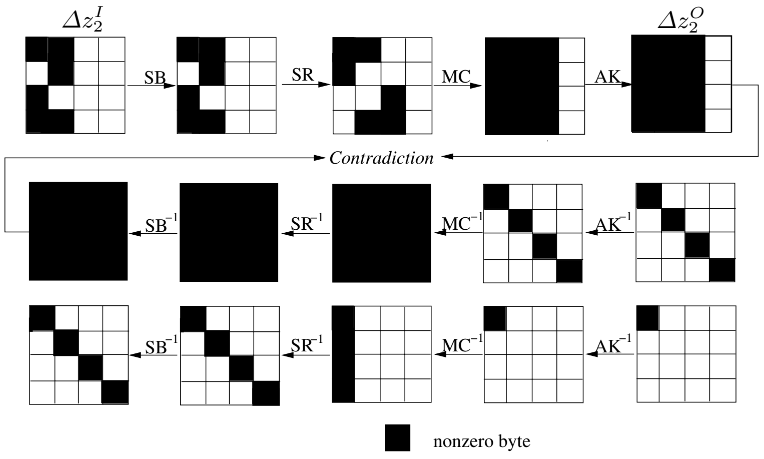


Fig. 4. 3-Round Impossible Differential Path of AES

used as a chaining or auxiliary permutation function, whose output is concealed by the final full AES encryption, as well as the output difference.

To get over this obstacle, we take advantage of the idea described in [19,20,21]. First, randomly choose two structures of messages, with the message differences of some specific forms. One example is that there is only one nonzero byte in the difference of the last word. The concrete structures are constructed based on the concrete MAC constructions. Second, utilize the birthday attack to search collisions between the two structures. Finally, once enough message collisions are collected, we can sieved the correct subkeys in the similar manner as the impossible differential cryptanalysis of block ciphers. The details of collecting collision pairs will be given in the next section.

9 Impossible Differential Cryptanalysis of Pelican and PC-MAC-AES

In this section, we present the impossible differential attacks based on the 3-round impossible differential path proposed in Section 8.

9.1 Internal State Recovery of Pelican

This subsection describes the internal state recovery attack on Pelican with one additional round at the beginning of the 3-round impossible differential path. The recovery of the internal state results in the derivation of an equivalent subkey, i. e., the state $y_0 = E_K(0)$. We depict the Pelican algorithm with two message words in Fig. 5 for simplicity.

We first consider the situation that there is no truncation at the final output, i. e., $l_m = 128$. From Fig. 5, we can see that a collision at C indicates a collision

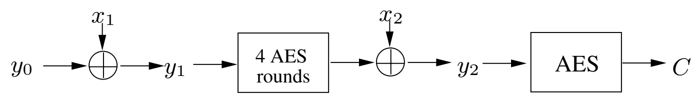


Fig. 5. PELICAN ($b = 2$)

at y_2 since the final AES encryption is a permutation. Because

$$y_2 = AES^{4r}(y_1) \oplus x_2,$$

where AES^{4r} stands for the 4-round AES, the inner collision at y_2 happens if and only if

$$AES^{4r}(y_1) \oplus x_2 = AES^{4r}(y_1') \oplus x_2',$$

which yields the output difference of 4-round AES

$$AES^{4r}(y_1) \oplus AES^{4r}(y_1') = x_2 \oplus x_2'. \tag{8}$$

If there is truncation at the final output, i. e., $l_m < 128$, then we need to distinguish the collision caused by inner collision, which means to detect the message pairs leading to $y_2 = y_2'$. Suppose $(x_1\|x_2, x_1'\|x_2')$ is a collision. Query the MAC with $(x_1\|x_2', x_1'\|x_2)$. If they still collide, we conclude that the pair $(x_1\|x_2, x_1'\|x_2')$ satisfies $y_2 = y_2'$, i. e., Eq. (8).

It is clear that, once an inner collision is detected, we can deduce the information of the output difference of the inner 4-round AES from Δx_2, and apply the impossible differential cryptanalysis. Therefore, it is essential to collect enough message pairs which cause inner collisions at y_2.

Message Pairs Collection Phase

We sieve the message pairs resulting in the inner collisions as follows.

1. Construct two structures, each has 2^{64} two-word messages. Randomly choose $(x_{1,2}, \ldots, x_{1,14})$, which are the bytes of the first word x_1 indexed by $(2, 3, 4, 7, 8, 9, 13, 14)$, and set the corresponding bytes of x_1' with the same values; randomly choose two 128-bit message words x_2 and x_2', with only one nonzero byte in $\Delta x_2 = x_2 \oplus x_2'$. The two structures are

$$S_1 = \{(x_1, x_2)|(x_{1,0}, x_{1,1}, x_{1,5}, x_{1,6}, x_{1,10}, x_{1,11}, x_{1,12}, x_{1,15}) \in \{0,1\}^{64}\},$$
$$S_2 = \{(x_1', x_2')|(x_{1,0}', x_{1,1}', x_{1,5}', x_{1,6}', x_{1,10}', x_{1,11}', x_{1,12}', x_{1,15}') \in \{0,1\}^{64}\}.$$

   It is noted that the difference Δx_1 is zero at bytes indexed by $(2, 3, 4, 7, 8, 9, 13, 14)$, where $\Delta x_1 = x_1 \oplus x_1'$.
2. Query MAC on the two structures, and search collisions between the corresponding MAC values of the two structures by the birthday attack.
   – If there is no truncation at the final output, the corresponding colliding message pairs cause inner collisions at y_2.

– Else, for all collected colliding pairs $(x_1\|x_2, x'_1\|x'_2)$, query the MAC on $(x_1\|x'_2, x'_1\|x_2)$. If still collide, $(x_1\|x_2, x'_1\|x'_2)$ must be an inner collision.

Since there are 2^{64} elements in each structure, and the internal state is 128-bit, one inner collision is expected to be found with probability 2^{-1}. Repeat the message pairs collection phase by choosing different $(x_{1,2}, x_{1,3}, x_{1,4}, x_{1,7}, x_{1,8}, x_{1,9}, x_{1,13}, x_{1,14})$, one inner collision pair is expected to be obtained. This means that, we can get one useful pair with about $2 \times 2 \times 2^{64} = 2^{66}$ chosen messages. To obtain 2^a colliding pairs, $2^a \times 2^{66} = 2^{a+66}$ chosen messages are required. Thus, the time complexity is 2^{a+66} queries.

For each collected pair, there is only one nonzero byte in Δz_4^O since there is only one nonzero byte in Δx_2, where $z_4^O = AES^{4r}(y_1)$. The input to the 4-round AES, y_1, equals to $x_1 \oplus y_0$, and the round subkeys are set to zero, so y_0 can be regarded as the subkey XORed before the first round, and is recovered in a similar manner as the impossible differential cryptanalysis of AES.

Internal State Recovery Phase
We can recover 8 bytes of y_0 at position $(0, 1, 5, 6, 10, 11, 12, 15)$ by exhaustive search directly (See Fig. 6).

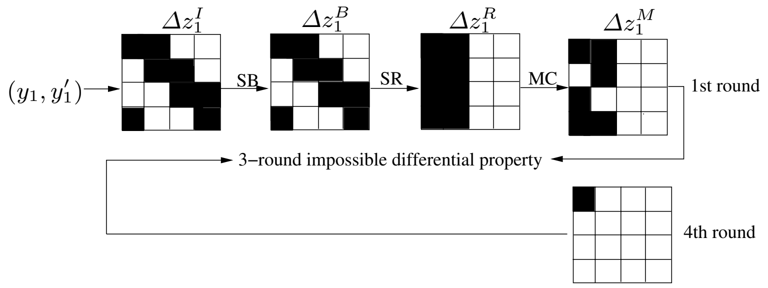


Fig. 6. Internal State Recovery of PELICAN

1. Initialize a list L to store the 2^{64} possible values $(y_{0,0}, y_{0,1}, y_{0,5}, y_{0,6}, y_{0,10}, y_{0,11}, y_{0,12}, y_{0,15})$.
2. For each of the 2^a valid pairs, perform partial encryption with each element in L, and obtain the first two columns of z_1^M and $z_1^{M\prime}$, respectively. From the fact that Δx_1 is zero at bytes $(2, 3, 4, 7, 8, 9, 13, 14)$, we can deduce that the last two columns of Δz_1^M are zero. Thus, if Δz_1^M is in the form of Δz_2^I as described in Property 1, the corresponding 8 bytes of y_0 must be wrong, because of Property 1. We delete it from the list L.

   After all pairs are processed, we expect that there is only one element in the list L, which is the correct one.

For random $(y_{0,0}, y_{0,1}, y_{0,5}, y_{0,6}, y_{0,10}, y_{0,11}, y_{0,12}, y_{0,15})$, the probability that Δz_1^M has the impossible form is $4 \cdot 2^{-16} = 2^{-14}$, since for the two zero bytes in the first two columns, there are 4 possible positions. Therefore, for each collected pair, we can filter out $2^{64} \cdot 2^{-14} = 2^{50}$ wrong $(y_{0,0}, y_{0,1}, y_{0,5}, y_{0,6}, y_{0,10}, y_{0,11}, y_{0,12}, y_{0,15})$, and one wrong value remains in list L with probability $1 - \frac{2^{50}}{2^{64}}$. After analyzing all 2^a pairs, the expected number of wrong elements left in L should satisfy

$$2^{64} \cdot (1 - \frac{2^{50}}{2^{64}})^{2^a} < 1.$$

This relation holds if we take $a = 2^{19.5}$.

In this manner, we can recover 8 bytes of the internal state y_0, and the other 8 bytes can be recovered in a similar way.

Complexity Estimation. For the message pairs collection phase, the data complexity is $2^{a+66} = 2^{85.5}$ chosen messages, and the time complexity is $2^{85.5}$ queries. For the internal state recovery phase, the time complexity is at most $2^{19.5} \cdot 2^{64} = 2^{83.5}$ one-round encryptions since there are $2^{19.5}$ collected pairs. Therefore, the total complexity is dominated by the message pairs collection phase, which is about $2^{85.5}$ queries and $2^{85.5}$ messages.

Selective Forgery Attack. Once the attacker obtains the value of the internal state y_0, he has full control of the internal state, and can create arbitrary colliding messages by calculating a proper 128-bit injection at the end.

9.2 Key Recovery Attack on PC-MAC-AES

The situation becomes a little different when it comes to PC-MAC-AES, where the simplified 4-round AES is applied after the second block, and there are two secret keys (K, L) involved in the MAC computation. We can use the divide-and-conquer technique to recover the two secret keys. The PC-MAC-AES with three message words is illustrated in Fig. 7.

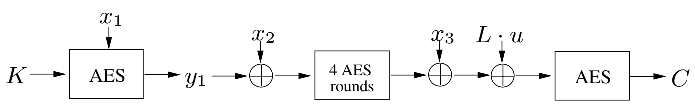


Fig. 7. PC-MAC-AES with Three Message Words

We proceed the key recovery attack according to the following procedure.

1. Construct two structures by prepending a fixed x_1 to each message of structures S_1 and S_2 given in Section 9.1. Randomly choose x_1, set the bytes at $(2, 3, 4, 7, 8, 9, 13, 14)$ of x_2 and x_2' to the same values, and choose two 128-bit message blocks x_3 and x_3' with only one nonzero byte in Δx_3. The following are the two structures, each has 2^{64} elements:

$$S_1' = \{(x_1, x_2, x_3) \mid (x_{2,0}, x_{2,1}, x_{2,5}, x_{2,6}, x_{2,10}, x_{2,11}, x_{2,12}, x_{2,15}) \in \{0,1\}^{64}\},$$
$$S_2' = \{(x_1, x_2', x_3') \mid (x_{2,0}', x_{2,1}', x_{2,5}', x_{2,6}', x_{2,10}', x_{2,11}', x_{2,12}', x_{2,15}') \in \{0,1\}^{64}\}.$$

2. Recover the value y_1 using the internal state recovery attack presented in Section 9.1. It is noted that, x_1 is unchanged when we choose different structures to collect enough colliding pairs.
3. Since $y_1 = E_K(x_1)$, K is recovered by exhaustive search directly.
4. When K is recovered, exhaustively search 2^{128} possibilities of L, and only the correct one is suggested by the MAC value C.

Complexity Estimation. The data complexity is the same as the internal state recovery attack on Pelican, which is about $2^{85.5}$ chosen messages, and the time complexity is dominated by the exhaustive search of the secret key, which is about 2^{128} queries, much lower than the 2^{256} security bound.

We note that even two keys are involved in PC-MAC-AES, the security of the algorithm does not get enhanced.

10 Conclusion

In this part, we adopt the techniques of detecting the inner near-collisions with some specific differences [19,20,21] to implement impossible differential cryptanalysis on Pelican, MT-MAC-AES and PC-MAC-AES, and all of them take the 4-round AES as the iteration function. Based on a 3-round impossible differential path of AES, we can recover the internal state of Pelican, which is an equivalent subkey, and the recovery leads to a selective forgery attack. The data complexity is $2^{85.5}$ chosen messages, and the time complexity is $2^{85.5}$ queries. This attack is applicable to MT-MAC-AES and PC-MAC-AES directly. For MT-MAC-AES, it turns to be a subkey recovery attack with the same complexity. For PC-MAC-AES, we can deduce the two secret keys separately with 2^{128} queries and $2^{85.5}$ chosen messages. Our attacks have a complexity greater than the birthday paradox, so they are not covered by the designers proofs.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments on the two parts.

References

1. Bahrak, B., Aref, M.R.: Impossible Differential Attack on Seven-Round AES-128. IET Information Security 2(2), 28–32 (2008)
2. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
3. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
4. Biham, E., Keller, N.: Cryptanalysis of Reduced Variants of Rijndael. In: 3rd AES Conference (2000)

5. Biryukov, A., Bogdanov, A., Khovratovich, D., Kasper, T.: Collision Attacks on AES-Based MAC: Alpha-MAC. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 166–180. Springer, Heidelberg (2007)
6. Boesgaard, M., Christensen, T., Zenner, E.: Badger - A Fast and Provably Secure MAC. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 176–191. Springer, Heidelberg (2005)
7. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. In: The First Advanced Encryption Standard Candidate Conference. NIST AES Proposal (1998)
8. Daemen, J., Rijmen, V.: A New MAC Construction Alred and A Specific Instance Alpha-MAC. In: Gilber, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 1–17. Springer, Heidelberg (2005)
9. Daemen, J., Rijmen, V.: The PELICAN MAC Function. IACR ePrint Archive (2005), `http://eprint.iacr.org/2005/088`
10. Huang, J., Seberry, J., Susilo, W.: On the Internal Structure of Alpha-MAC. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 271–285. Springer, Heidelberg (2006)
11. ISO/IEC 9797-1, Information technology - Security Techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using A Block Cipher, ISO (1999)
12. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
13. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0, and SHA-1. In: Prisco, R.D., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
14. Kurosawa, K., Iwata, T.: TMAC: Two-Key CBC MAC. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 33–49. Springer, Heidelberg (2003)
15. Minematsu, K., Tsunoom, Y.: Provably Secure MACs from Differentially-Uniform Permutations and AES-Based Implementations. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 226–241. Springer, Heidelberg (2006)
16. Phan, R.C.-W.: Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES). Information Processing Letters 91(1), 33–38 (2004)
17. Preneel, B., van Oorschot, P.: MDx-MAC and Building Fast MACs from Hash Functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
18. Wang, W., Wang, X., Xu, G.: Impossible Differential Cryptanalysis of Pelican, MT-MAC-AES and PC-MAC-AES, Cryptology ePrint Archive, Report 2009/005 (2009), `http://eprint.iacr.org/2009/005`
19. Wang, X., Wang, W., Jia, K., Wang, M.: New Distinguishing Attack on MAC using Secret-Prefix Method. In: FSE 2009 (to appear, 2009)
20. Wang, X., Yu, H., Wang, W., Zhang, H., Zhan, T.: Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 121–133. Springer, Heidelberg (2009)
21. Yuan, Z., Jia, K., Wang, W., Wang, X.: Distinguishing and Forgery Attacks on Alred and Its AES-based Instance Alpha-MAC. Cryptology ePrint Archive, Report 2008/516 (2008), `http://eprint.iacr.org/2008/516`
22. Yuval, G.: How to Swindle Rabin. Cryptologia 3, 187–189 (1979)

Distinguisher and Related-Key Attack on the Full AES-256

Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić

University of Luxembourg
{alex.biryukov,dmitry.khovratovich,ivica.nikolic@uni.lu}

Abstract. In this paper we construct a chosen-key distinguisher and a related-key attack on the full 256-bit key AES. We define a notion of *differential q-multicollision* and show that for AES-256 q-multicollisions can be constructed in time $q \cdot 2^{67}$ and with negligible memory, while we prove that the same task for an ideal cipher of the same block size would require at least $O(q \cdot 2^{\frac{q-1}{q+1}128})$ time. Using similar approach and with the same complexity we can also construct q-pseudo collisions for AES-256 in Davies-Meyer mode, a scheme which is provably secure in the ideal-cipher model. We have also computed partial q-multicollisions in time $q \cdot 2^{37}$ on a PC to verify our results. These results show that AES-256 can not model an ideal cipher in theoretical constructions. Finally we extend our results to find the first publicly known attack on the full 14-round AES-256: a related-key distinguisher which works for one out of every 2^{35} keys with 2^{120} data and time complexity and negligible memory. This distinguisher is translated into a key-recovery attack with total complexity of 2^{131} time and 2^{65} memory.

Keywords: AES, related-key attack, chosen key distinguisher, Davies-Meyer, ideal cipher.

1 Introduction

The Advanced Encryption Standard (AES) is a block cipher which was chosen by NIST from a set of 15 candidate designs in a thorough evaluation process that lasted from September 1997 till October 2000. On November 26, 2001 Rijndael [5], a 128-bit block, 128/192/256-bit key block cipher has become a standard as U.S. FIPS 197 [12]. In June 2003 the US government has approved the use of 128, 192, 256 bit key AES for SECRET and 192, 256-bit key AES for TOP SECRET information [13]. In the last ten years AES has been subject to very intensive cryptanalytic effort, with best currently known attacks breaking 7, 10, 10 rounds for respective keysizes (128, 192, 256), with very high complexities.

In this paper we show for the first time in the open literature distinguishers and related-key attacks on the full 14-round 256-bit key AES. Research presented in this paper follows the logic described in Fig. 1. First we identified slow diffusion and other differential weaknesses in the key schedule of AES-256 which match

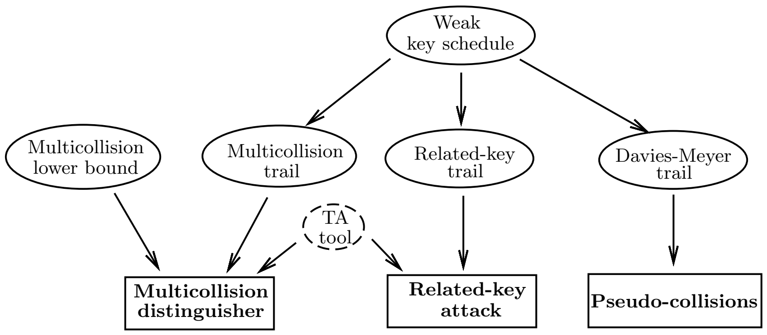


Fig. 1. Outline of the research presented in this paper

nicely with the differential properties of the round function. This allows us to construct local collisions for AES, i.e. two round difference propagation patterns which result in low weight difference in the subkeys and zero difference in the 128-bit block. We concatenate four such local collisions together and add another 6-round trail on top in order to cover full 14 rounds of AES-256. The trail[1] has 41 active S-boxes (36 in the block and 5 in the key schedule), so we apply a special tool, a triangulation algorithm (designed for the purpose of finding collisions in hash functions), in order to find keys and plaintexts that conform to the trail.

From this point we go in two directions. First we show that for AES-256 one can construct a *chosen-key* distinguisher based on the new notion of a *differential q-multicollision* in time $q \cdot 2^{67}$ and with negligible memory. We prove that the same task for an ideal cipher of the same block size would require at least $q \cdot 2^{\frac{q-1}{q+1}128}$ time for $q \leq 57$ and at least $q \cdot 2^{\frac{q-2}{q+2}128}$ for $q > 57$. I.e. for $q > 3$ the differential multicollision for AES-256 can be constructed significantly faster than for an ideal cipher. Previously a *known-key* distinguisher for seven rounds of AES with 2^{56} texts was found in [11]. To verify our results we found partial q-multicolisions in several hours on a PC using the publicly available implementation of AES-256.

As a direct application of this differential q-multicollision distinguisher we show that AES-256 when used in the Davies-Meyer mode allows to construct q pseudo-collisions with *fixed* differences ΔIV, ΔM in the IV and the message with complexity $q \cdot 2^{67}$. Again, such a result would require at least $q \cdot 2^{\frac{q-2}{q+2}128}$ time for the ideal cipher in the Davies-Meyer mode. Results of this type try to enchance our definitions of block cipher security and to fill the gap between theoretical models like random oracle and ideal cipher and the real world of ciphers which have fixed description and are efficiently computable [2,4]. However a proper security definition which would capture the intuition behind chosen/known key attacks is still an open problem.

[1] We use colors in the diagrams of the trails, so please refer also to the tables in the appendix if you print this paper on a black and white printer.

Table 1. Best attacks on AES-256

Attack	# rounds	# keys	Data	Time	Memory	Source
Known-key integral	7	1	2^{56}	2^{56}	2^{56}	[11]
Partial sums	9	256	2^{85}	2^{226}	2^{32}	[6]
Related-key rectangle	10	64	2^{114}	2^{173}	?	[1,10]
q-multicollisions	14	$2q$	$2q$	$q \cdot 2^{67}$	-	Sec. 2
Partial q-multicollisions	14	$2q$	$2q$	$q \cdot 2^{37}$	-	Sec. 2.3
Related-key distinguisher	14	2^{35}	2^{119} *	2^{119} *	-	Sec. 4.1
Related-key key recovery	14	2^{35}	2^{96} *	2^{96} *	2^{65}	Sec. 4.2

* — for each key.

The second direction that we studied was application of the trails that we have found to more standard attacks on a block cipher, for example related-key attacks. In particular we show that by changing the top two rounds of the trail that we used previously one obtains a differential trail with only 24 active S-boxes (19 in the round function and 5 in the key schedule). From this trail we can construct a differential distinguisher for AES-256 which works for one key out of 2^{35} and has complexity 2^{120} data and time, and negligible memory. This distinguisher can be used to mount a key-recovery attack on AES-256 with total complexity of $2^{35+96} = 2^{131}$ time and 2^{65} memory. We summarize our findings in Table 1.

This paper is organized as follows: in Section 2 we prove a lower bound on the complexity of finding differential q-multicollisions in the case of an ideal cipher and construct a distinguisher for the full AES-256. In Section 3 we show an application of these results to finding pseudo-collisions for the Davies-Meyer hashing mode instantiated with AES-256. In Section 4 we show a related-key attack on the full AES-256. In Section 5 we discuss new design criteria for block cipher key schedule as a consequence of our attack. Section 6 concludes the paper. In Appendix A we provide technical details about our differential trails.

Discussion. It is clear that the *open key* (chosen or known) security model is new and is still lacking a proper security definition. However we think that if one can support an open-key attack with a proof of security against such attack for the ideal cipher, this gives additional confidence that such property (in our case "differential multicollisions") should not be present in a good cipher. There are many constructions provably secure [3,7] in the ideal cipher model. This model assumes that both the key and the plaintext are accessible to the attacker. If a block cipher (e.g., AES) exhibits a property that should not appear in the ideal cipher then instantiation of a provably secure construction with this cipher could undesirably weaken the construction. Our Davies-Meyer example is exactly to show that a construction provably secure in the ICM can break down if instantiated with AES-256 (such a hash function was never proposed for another reason — 128-bit state is too short for a modern hash). The fact that this property does not automatically carry on to Davies-Meyer instantiated with all the other block-ciphers (hopefully), shows a non-trivial weakness of AES-256.

2 Multicollision Distinguisher

In this section we provide a chosen-key distinguisher for AES-256 which has practical complexity.

Definition 1. *A set of two differences and q pairs*

$$\{\Delta_K, \Delta_P;\ (P_1, K_1),\ (P_2, K_2), \ldots, (P_q, K_q))\}$$

is called a differential q-multicollision *for a cipher $E_K(\cdot)$ if*

$$\begin{aligned} E_{K_1}(P_1) \oplus E_{K_1 \oplus \Delta_K}(P_1 \oplus \Delta_P) = E_{K_2}(P_2) \oplus E_{K_2 \oplus \Delta_K}(P_2 \oplus \Delta_P) = \\ = \cdots = E_{K_q}(P_q) \oplus E_{K_q \oplus \Delta_K}(P_q \oplus \Delta_P). \quad (1) \end{aligned}$$

A differential q-multicollision can be also viewed as a set of q right pairs with respect to the related-key differential, where the key is not fixed.

We compare the task of constructing a differential q-multicollision for an ideal cipher with that for AES-256. This task for an ideal cipher, i.e. a set of 2^k randomly chosen permutations, would require treating it as a black-box and making only encryption/decryption queries. We expect that for a good cipher with no (yet discovered) structural flaws, the task of constructing a differential q-multicollision would have the same complexity as for an ideal cipher.

Let us compute this complexity measured in the number of queries. Since the cipher is ideal, an adversary is only given an access to the encryption and decryption oracles, both having two inputs (a key and a plaintext/ciphertext) and one output. This is the same model of an adversary as in [2,3, p. 329].

In the beginning no triplet $\langle$*plaintext*, *key*, *ciphertext*$\rangle$ is defined. Then, for each query of the adversary "$E_K(P) = ?$" the encryption oracle takes a random value C from a possible range (where $E_K(\cdot)$ is yet undefined) and thus defines $E_K(P) = C$. Also $E_K^{-1}(C)$ becomes defined. The same rule holds for a decryption query.

Lemma 1. *To construct a differential q-multicollision for an ideal cipher with an n-bit block an adversary needs at least $O(q \cdot 2^{\frac{q-2}{q+2}n})$ queries on the average.*

Proof. See Sec. 2.1.

Remark 1. For small q, when the lower bound does not exceed 2^{n-1}, a better estimate is obtained (see the proof of the lemma). In our case, for $n = 128$, an adversary needs at least $q \cdot 2^{\frac{q-1}{q+1}128}$ queries if $q \leq 57$.

Surprisingly, differential multicollisions for AES-256 can be constructed substantially faster. Furthermore, we can set $\Delta_P = 0$ in a multicollision, so a stronger statement holds.

Theorem 1. *A differential q-multicollision with $\Delta_P = 0$ for AES-256 can be found with time complexity $q \cdot 2^{67}$.*

Proof. See Sec. 2.2.

Thus for $q > 3$ a differential q-multicollision for AES-256 can be constructed significantly faster than for an ideal cipher.[2] Therefore, AES-256 can not model an ideal cipher.

2.1 Proof of Lemma 1

Proof. Let A be an adversary attacking the cipher, and assume that A asks its oracles a total of L queries, where $L < 2^{n-1}$. Assume that a multicollision of the form (1) is found. Let us compute the probability of this event. First, we rewrite (1) as $U_1 = U_2 = \cdots = U_q$. With each term $U_j = E_{K_j}(P_j) \oplus E_{K_j \oplus \Delta_K}(P_j \oplus \Delta_P)$ we associate an integer t_j such that t_j-th oracle query determines the value of U_j, i.e., computes the last (chronologically) element of the sum. Without loss of generality, assume that $t_1 < t_2 < \cdots < t_q$. Finally, define t'_1 as the index of the query that determines the first element of the sum U_1.

$$\overbrace{\underbrace{E_{K_1}(P_1)}_{\text{queried at } t'_1} \oplus \underbrace{E_{K_1 \oplus \Delta_K}(P_1 \oplus \Delta_P)}_{\text{queried at } t_1}}^{U_1} = \overbrace{\underbrace{E_{K_2}(P_2)}_{\text{queried before } t_2} \oplus \underbrace{E_{K_2 \oplus \Delta_K}(P_2 \oplus \Delta_P)}_{\text{queried at } t_2}}^{U_2} =$$
$$= \cdots = \overbrace{\underbrace{E_{K_q}(P_q)}_{\text{queried before } t_q} \oplus \underbrace{E_{K_q \oplus \Delta_K}(P_q \oplus \Delta_P)}_{\text{queried at } t_q}}^{U_q}. \quad (2)$$

Now compute for every $(t'_1, t_1, t_2, t_3, \ldots, t_q)$ the probability that this set defines a differential q-multicollision. Before submitting t_i-th query, $i > 1$, the following equation holds:

$$U_1 = U_2 = \cdots = U_{i-1},$$

where terms of $U_1, U_2, \ldots, U_{i-1}$ are completely determined by a tuple $(t'_1,\ t_1,\ t_2,\ t_3,\ \ldots,\ t_{i-1})$. Indeed, from t'_1 and t_1 we define K_1, Δ_K, P_1, Δ_P; from t_j we define K_j and P_j.

Just before the moment t_i only one term of U_i is computed — w.l.o.g. let it be $E_{K_i}(P_i)$. Thus the equality $U_{i-1} = U_i$ should hold, i.e.

$$U_{i-1} = E_{K_i}(P_i) \oplus \underbrace{E_{K_i \oplus \Delta_K}(P_i \oplus \Delta_P)}_{\text{queried at } t_i}$$

[2] Moreover, even for $q = 3$ we are not aware of any algorithm faster than $2^{2n/3}$.

By our definition, t_i is the first moment when $E_{K_i \oplus \Delta_K}(P_i \oplus \Delta_P)$ is queried. Then either the decryption or the encryption oracle is called. In the first case the decryption oracle is called with a ciphertext C and a key K, which for some i should be equal to $K_i \oplus \Delta_K$. By the definition of t_i, the value C is chosen from the set where $E_{K_i \oplus \Delta_K}(\cdot)$ is undefined. To become a part of a multicollision, there should exist P_i such that $C = E_{K_i}(P_i) \oplus U_{i-1}$. On the other hand, after the decryption oracle is called, the following equation should hold:

$$E^{-1}_{K_i \oplus \Delta_K}(C) = P_i \oplus \Delta_P. \tag{3}$$

Since $L < 2^{n-1}$, not more than 2^{n-1} texts were encrypted or decrypted with the key $K_i \oplus \Delta_K$. So the probability that (3) holds does not exceed $1/2^{n-1}$.

In the second case, let the encryption oracle be queried with a plaintext P and a key K, which for some i should be equal to $K_i \oplus \Delta_K$. For an answer C, a similar equation should hold:

$$C = U_{i-1} \oplus E_{K_i}(P_i). \tag{4}$$

The same probability argument holds for this equation. Therefore, for every $i \geq 2$ we get a multiplier 2^{1-n} to the probability that a tuple $(t'_1, t_1, t_2, t_3, \dots, t_q)$ defines a differential q-multicollision. There are $\binom{L}{q+1}$ such tuples, each defining a differential q-multicollision with probability at max $2^{(q-1)(1-n)}$. We get the following equation for the number of queries required to get a q-multicollision with probability 1/2:

$$\binom{L}{q+1} \geq 2^{(q-1)(n-1)-1}. \tag{5}$$

Let us simplify the left part:

$$\binom{L}{q+1} = \frac{L!}{(L-q-1)!(q+1)!} = \frac{L(L-1)\cdots(L-q)}{(q+1)!} \leq \\ \leq \frac{L^{q+1}}{(q+1)!} \leq \frac{L^{q+1}}{\frac{(q+1)^{q+1}}{e^{q+1}}} = \left(\frac{eL}{q+1}\right)^{q+1}. \tag{6}$$

Substitute the result to (5):

$$\left(\frac{eL}{q+1}\right)^{q+1} \geq 2^{(q-1)(n-1)-1} \Rightarrow L \geq \frac{q+1}{e} 2^{\frac{q-1}{q+1}(n-1)-1} = O(q \cdot 2^{\frac{q-1}{q+1}n}). \tag{7}$$

This is the bound for the number of queries needed to construct a multicollision with probability 1/2. By Markov's inequality, the average number of queries exceeds this bound divided by two, so the right part of (7) is still a correct lower bound.

Now consider the case when $L \geq 2^{n-1}$. Let $\overline{K}$ be the set of keys such that there were more than 2^{n-1} encryption or decryption queries on each of these keys. Define $l = |\overline{K}|$. If $l > q-2$ then L exceeds $q \cdot 2^{n-2}$, which implies the statement of the lemma. If $l \leq q-2$ then there are at least $q-l$ sums U_i in (2) that do not involve keys from $\overline{K}$. So if a q-multicollision has been found, then a $(q-l)$-multicollision has too been found such that it does not involve keys from $\overline{K}$. Then all the arguments on the probability of this event can be carried out from the first part of the proof.

Therefore, we gets the following inequality on L:

$$L \geq \frac{q-l+1}{e} 2^{\frac{q-l-1}{q-l+1}(n-1)-1} + l \cdot 2^{n-1}.$$

For $l < q/2$ we get the following:

$$L \geq \frac{q}{2e} 2^{\frac{q-2}{q+2}(n-1)-1} = O(q \cdot 2^{\frac{q-2}{q+2}n}). \tag{8}$$

For $l \geq q/2$ we get

$$L \geq q \cdot 2^{n-2} = O(q \cdot 2^n). \tag{9}$$

Equations (7), (8), and (9) complete the proof. □

Remark 2. The function $F_{\Delta_K,\Delta_P}(K,P) = E_K(P) \oplus E_{K \oplus \Delta_K}(P \oplus \Delta_P)$ is a xor of two permutations. Patarin in [14] has shown that the xor of two random permutations can not be distinguished from a pseudo-random function with less than 2^n queries. In [15] it was proven that q-multicollision search for a random function requires at least $(q!)^{\frac{1}{q}} 2^{\frac{q-1}{q}n}$ effort. In our case we can not use this result since it assumes that Δ_K and Δ_P are fixed in advance while we allow an attacker to choose them during the attack.

2.2 Proof of Theorem 1

Here we construct a differential q-multicollision (1) with $\Delta_P = 0$ in $q \cdot 2^{67}$ time. This is done in 5 steps:

1. Build a differential trail, which is efficient for the multicollision search.
2. Derive Δ_K from the trail.
3. Choose the active S-boxes, whose inputs will be fixed in the triangulation algorithm. Denote this set by S.
4. Run the triangulation algorithm and derive a set of free variables.
5. Produce q pairs (P,K) for (1) as follows:
   (a) Assign inputs to S-boxes from S with admissible values.
   (b) Assign free variables randomly.
   (c) Produce (P,K).
   (d) Check if (P,K) and $(P,K \oplus \Delta_K)$ fit (1).

We expect that most of our readers are familiar with the description of AES and thus point out only main features of AES-256 that are crucial for our proof.

Differential Trail

Notations. Differential trail (also called differential characteristic) is a sequence of differences in all the internal states of the cipher and all the subkeys. If we distinguish only between zero and non-zero (byte) differences, we call such a trail *a trail with truncated differences.*

We denote the subkey of round i by K^i, i.e. the first (whitening) subkey is K^0, the subkey of round 1 is K^1, etc., the last subkey is K^{14}. The difference in K^i is denoted by ΔK^i. Bytes of a subkey are denoted by $K^l_{i,j}$, where i stands for the row index, and j stands for the column index in the standard matrix representation of AES. Bytes of the plaintext are denoted by $P_{i,j}$, and bytes of the internal state after the SubBytes transformation in round r are denoted by $A^r_{i,j}$. Let also $B^r_{i,j}$ denote a byte in position (i,j) after the r-th application of MixColumns.

Features of AES-256. AES-256 has 14 rounds and a 256-bit key, which is two times larger than the internal state. Thus the key schedule consists of only 7 rounds. One key schedule round consists of the following transformations:

$$\begin{aligned}
K_{i,0} &\leftarrow S(K_{i+1,7}) \oplus K_{i,0} \oplus C_r, && 0 \le i \le 3;\\
K_{i,j} &\leftarrow K_{i,j-1} \oplus K_{i,j}, && 0 \le i \le 3,\ 1 \le j \le 3;\\
K_{i,4} &\leftarrow S(K_{i,3}) \oplus K_{i,4}, && 0 \le i \le 3;\\
K_{i,j} &\leftarrow K_{i,j-1} \oplus K_{i,j}, && 0 \le i \le 3,\ 5 \le j \le 7,
\end{aligned} \tag{10}$$

where $S()$ stands for S-box, and C_r — for the round-dependant constant. Therefore, every round has 8 S-boxes.

Weakness in the key schedule. Two features of the key schedule help us to build a good differential trail. First, the key schedule has a slow diffusion in backward direction. It means that a difference in a single byte $K_{0,0}$ will propagate to only two bytes, $K_{0,0}$ and $K_{0,1}$, if we apply the inversion of the key schedule round. The next inverted round will affect only one more byte, etc. Thus we can build a trail with a low-weight difference in key schedule if we start with a low-weight difference in the last round and then step backwards.

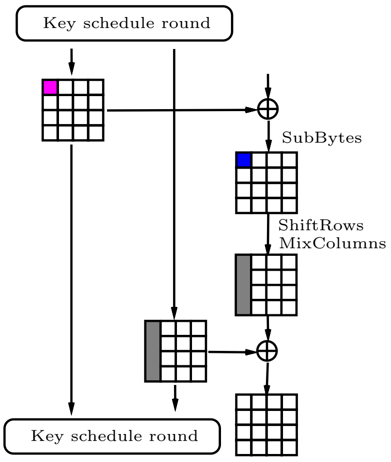


Fig. 2. A local collision

The second feature is unique to AES-256 due to its "key size"/"state size" ratio, $N_k/N_b = 2$. We can inject "good" values with the first part of the key and then cancel them, after they pass the round, with the second part of the key. We call this a *local collision* (Fig. 2).

Constructing a trail. Step by step, we construct a differential trail from the last rounds to the first ones. The trail is described in details in Appendix A, both in a truncated form and with the actual differences given. The trail has 41 active S-boxes, and 5 of them are in the key schedule.

Search for a solution. After the trail has been defined, we produce a pair (P, K) which with pair $(P, K + \Delta_K)$ fits the trail and thus is a part of a differential multicollision (1) since all such pairs have the same difference in the ciphertext. Now note that the trail explicitly states all the non-zero input δ_I and output δ_O differences of the S-boxes. According to the S-box properties, there are at most 4 solutions of the equation $S(x \oplus \delta_I) \oplus S(x) = \delta_O$. This set of solutions we call *admissible inputs*. Therefore, (P, K) and $(P, K + \Delta_K)$ fit the trail if and only if in the execution $E_K(P)$ all active S-boxes get admissible inputs. In the next paragraphs we explain how to construct such executions efficiently.

Triangulation Algorithm

Search for free variables. The triangulation algorithm was proposed in [9] as a tool for solving systems of non-linear equations, which appear in differential attacks. Given the constraints on the internal variables, the algorithm outputs a special set of variables, called *free variables*. These free variables can be assigned randomly; and this assignment together with pre-fixed variables completely and efficiently determines the whole execution. The fewer variables are fixed the better the algorithm works.

Our goal is to efficiently produce the cipher executions in which active S-boxes get admissible values. However, the algorithm can not process all the active S-boxes of our trail since they are positioned too far from one another.

We found that we can fix inputs to 30 out of 41 S-boxes: all the active S-boxes in the internal states of rounds 1–4 and all the 5 active S-boxes in the key scheduling. The triangulation algorithm outputs 18 free variables, out of these 11 are in the key and provide freedom in the choice of the key for the distinguisher. These free variables are listed below.

Key	Internal state
$K^1_{2,0}$, $K^1_{3,0}$, $K^1_{3,1}$, $K^2_{0,1}$, $K^3_{0,1}$	$A^2_{0,1}$, $B^2_{1,1}$, $B^2_{1,2}$
$K^3_{2,0}$, $K^3_{3,0}$, $K^3_{2,1}$, $K^4_{0,1}$, $K^5_{0,1}$, $K^5_{3,3}$	$B^2_{2,3}$, $B^2_{3,3}$, $A^3_{1,1}$, $A^3_{1,2}$

Constructing a pair. Having assigned 18 free variables randomly and 30 S-box inputs with an admissible value, we substitute these values to the equations, which have been ordered by the triangulation algorithm. One by one, all the variables are determined and thus a pair (P, K) is defined. It fits the trail if and only if the 11 S-boxes not covered by the triangulation algorithm get admissible values as inputs. For the S-box in round 6 only 2 values are admissible so the probability is 2^{-7} while for the other 10 S-boxes 4 values are admissible. This results in the overall probability $2^{-(7+10\cdot 6)} = 2^{-67}$. Thus out of 2^{67} pairs one fits the trail on average. We wrote a program and checked that the distribution of the pairs is random enough so the probability estimates are likely to be correct. We also checked experimentally that bottom 7 rounds of AES produce expected difference after 2^{30} pairs on the average, exactly as predicted by the trail.

The complexity of the attack. Recall the scheme of the attack, which was given in the beginning of the proof. The first four steps are precomputations and actually have negligible cost. The triangulation algorithm works less than a second. The last step requires only to substitute the values into the equations one by one, which is computationally equivalent to a single encryption. Thus to get a right pair we need about 2^{67} operations each equivalent to one encryption. The attack needs negligible memory and is fully parallelizable.

2.3 Practical Distinguisher

The definition of differential q-multicollision can be further relaxed if we allow arbitrary difference at some byte positions of Δ_P, Δ_K or Δ_C. Although an attacker gets more freedom, finding such a construction for an ideal cipher becomes easier as well. To get a lower bound, only a slight modification of Lemma 1 is required.

For the 13 rounds of AES-256 the complexity of finding this type of differential 5-multicollision, with fixed difference in 14 bytes of the plaintext and fixed ciphertext difference can be lower bounded by $2^{\frac{4\cdot 112}{6}} = 2^{74.6}$ computations. For the full AES-256 a differential 10-multicollision with half of the plaintext difference fixed, and the fixed ciphertext difference the lower bound is $2^{\frac{9\cdot 64}{11}} = 2^{52.3}$ computations. Note that these lower bounds are far from being tight. In practice we expect an efforts of 2^{112} and 2^{64} for finding *each* extra collision for 13 and 14 rounds of AES-256 respectively, since the differences are structured and *fixed.*

At the same time we can do it much faster, in just $q \cdot 2^{37}$ in both cases, which allowed us to compute these distinguishers in several hours on a PC. The actual values will be given in the extended version of this paper. The core of a practical distinguisher is a multicollision trail (Figure 5), where the behavior of S-boxes in the first two rounds is not restricted. Computing from the middle, we get 14 bytes with fixed difference before the second round, and 8 bytes with fixed difference before the first round. The triangulation algorithm covers all but six active S-boxes in rounds 3–14 so that we find a (partial) q-multicollision with complexity $q \cdot 2^{37}$.

3 Pseudo-collisions for AES-Based Hashing

The Davies-Meyer mode of blockcipher-based hashing has been proven collision-resistant if instantiated by an ideal cipher [3]. In this section we show a similar proof in the ideal-cipher model for the q pseudo-collision resistance, when differences in the IV and the message (Δ_I, Δ_M) are fixed. We then show that it is relatively easy to find q pseudo-collisions for AES-256 in the Davies-Meyer mode. We also point out that we construct one-block pseudo-collisions and thus the technique of Joux [8] does not apply here.

Our goal is for fixed differences Δ_I, Δ_M to find many pseudo-collisions for the $H^E(I, M) \stackrel{\text{def}}{=} E_M(I) \oplus I$ which is the Davies-Meyer compression function with AES-256 as the underlying cipher. Here I is the 128-bit IV, and M is a 256-bit message block. A pseudo-collision satisfies the following equality:

$$H^E(I, M) = H^E(I \oplus \Delta_I, M \oplus \Delta_M)$$

Let us rewrite it:

$$\begin{aligned} E_M(I) \oplus I = E_{M \oplus \Delta_M}(I \oplus \Delta_I) \oplus I \oplus \Delta_I \ \Leftrightarrow& \\ \Leftrightarrow\ E_M(I) \oplus E_{M \oplus \Delta_M}(I \oplus \Delta_I) = \Delta_I.& \quad (11) \end{aligned}$$

While finding many pseudo-collisions with different Δ_I, Δ_M can be done using the birthday paradox, the same task for fixed Δ_I, Δ_M is hard. This problem can be expressed as finding a solution of

$$\begin{aligned} E_{M_1}(P_1) \oplus E_{M_1 \oplus \Delta_M}(P_1 \oplus \Delta_I) = E_{M_2}(P_2) \oplus E_{M_2 \oplus \Delta_M}(P_2 \oplus \Delta_I) = \cdots =& \\ = E_{M_q}(P_q) \oplus E_{M_q \oplus \Delta_M}(P_q \oplus \Delta_I) = \Delta_I,& \quad (12) \end{aligned}$$

which is harder than finding a differential q-multicollision for $E_K(\cdot)$, because the ciphertext difference is unrestricted in (1). Therefore, Lemma 1 gives us a lower bound on the complexity of this attack.

Corollary 1. *To construct q pseudo-collisions (12) with fixed Δ_I, Δ_M for an ideal cipher with an n-bit block an adversary needs at least $O(q \cdot 2^{\frac{q-2}{q+2}n})$ queries on average.*

Theorem 2. *For AES-256 in the Davies-Meyer mode q pseudo-collisions (12) can be found in time $q \cdot 2^{67}$.*

Proof. Pseudo-collision attack on the Davies-Meyer mode requires the difference in the plaintext P to be equal to the difference in the ciphertext C. The Davies-Meyer feed-forward would then cancel this difference. Our differential trail needs only to be slightly modified for this purpose. The first round of the new trail is shown in Fig. 3 (the actual values are given in Appendix A); the other rounds are the same.

The resulting trail has 41 active S-boxes. The triangulation algorithm covers the same active S-boxes as in the proof of Theorem 1 and outputs 18 free variables. The complexity is thus the same.

Corollary 2. *AES-256 can not be used to instantiate the Davies-Meyer hashing mode.*

We expect that similar results can be shown for the other blockcipher-based constructions which are provably secure in the ideal-cipher model [3].

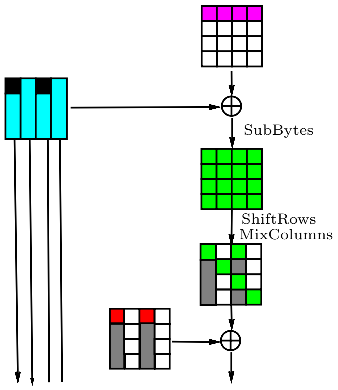


Fig. 3. The first round of the differential trail for the attack on AES-256 in the Davies-Meyer mode

4 Related-Key Attack on AES-256

In this section we demonstrate two results: a related-key distinguisher and a key-recovery attack based on this distinguisher for the full AES-256.

4.1 Distinguisher

The first two rounds of the multicollision trail can easily be modified to build a related-key distinguisher with relatively few active S-boxes — see Appendix A for the details of the trail. The resulting trail has 19 active S-boxes in the internal states. The difference propagates through 14 S-boxes with probability $2^{-6\cdot 14}$, and through the remaining five with prob. $2^{-7\cdot 5}$. Therefore, we get a distinguisher with probability $2^{-(14\cdot 6+5\cdot 7)} = 2^{-119}$. However, the trail has five additional active S-boxes (each with probability 2^{-7}) in the key schedule. As a result, the distinguisher works for 1 out of 2^{35} related-key pairs on the average.

4.2 Key Recovery

There are several ways how our trail can be used for the full 256-bit key recovery. Here we present one possibility. Steps of the attack are illustrated in Fig. 4 and are described below.

First step. We change the trail (see Fig. 4 or Appendix) to get more active S-boxes in the first two rounds which allows us to recover the key bytes at the entrance to these S-boxes. A new trail has eight active S-boxes in the first round in a “checkerboard pattern” and two in the second round. Our goal is to find ten key bytes $K^0_{0,0}$, $K^0_{0,2}$, $K^0_{1,1}$, $K^0_{1,3}$, $K^0_{2,0}$, $K^0_{2,2}$, $K^0_{3,1}$, $K^0_{3,3}$, $K^1_{0,0}$, $K^1_{0,2}$ so we execute the following procedure for each of the 2^{35} related-key pairs:

1. Repeat 2^{31} times:
   (a) Compose two structures of 2^{64} plaintexts as specified below.

   (b) Encrypt the 1st structure with K and the 2nd with $K \oplus \Delta_K$.
   (c) Sort the ciphertexts and check for a pair with the difference Δ_C.
   (d) Save a good pair if it is found.
2. For each candidate pair derive 2^{16} variants for the ten key bytes (see details below).
3. Pick the key candidate with the maximal number of votes.

The overall complexity of this procedure is $2^{31+65} = 2^{96}$ data and time, and 2^{64} memory, and it finds 80 bits of the key.

Each structure has all the possible values in bytes $\{P_{i,j}, (i+j) \mod 2 = 0\}$ (these bytes line up in columns 0 and 2 after the ShiftRows of the 1st round). The other bytes are random constants with the constraint that the two structures are related by Δ_P as in the distinguisher. For a fixed key, each structure contains 2^{64} pairs with the proper differences after the S-boxes of the 2nd round (yellow differences in two bytes). The remaining active S-boxes in rounds 3–14 require probability 2^{-93} for the trail to be fulfilled. Each structure produces a right pair of ciphertexts with probability $2^{64-93} = 2^{-29}$. Thus 2^{31} structures produce on average 4 right pairs and $2^{31+128-128} = 2^{31}$ wrong pairs.

Due to the uniform differential properties of AES S-boxes each active S-box for which we know input and output differences would suggest to us two candidates for the key byte of this S-box. For a candidate pair we guess two byte differences[3] at $B^1_{0,0}$ and $B^1_{0,2}$. We know the difference in the remaining three bytes of each column, since they should cancel out with the differences coming from the key. Thus we can undo the MixColumns for each column which allows us to know the output differences of eight S-boxes of the 1st round. We know the input differences for these S-boxes from the plaintext. In addition the two guessed difference bytes serve as inputs to the two active S-boxes of the 2nd round. The output differences for these S-boxes are known from the trail. Thus we have 10 S-boxes for which we know input and output differences which gives us $2^{10} \cdot 8 \cdot 8 = 2^{16}$ possibilities for 80-bits of the key per candidate pair. The 2^{31} wrong pairs would suggest $2^{16} \cdot 2^{31} = 2^{47}$ random keys, while the four good ones would all vote for the correct 80-bit key and some random keys. No wrong key guesses survive this step and we get 80 key bits as a result.

Second step. We proceed with changing top rounds of the trail to derive other key bytes. We remove the 2^{-6} condition on the input to the active S-box (0,0) of the 3rd round. Then we get five active S-boxes in the second round and 16 active S-boxes in the first round. We prepare 2^{90} pairs with the ciphertext difference as in the trail and decrypt them. We will try to detect pairs ($2^{90-87} = 8$ on the average) that pass the conditions in rounds 3–14. We partially encrypt all the resulting plaintext pairs using the known 80-bits and check whether the columns $\Delta B^1_{*,0}$ and $\Delta B^1_{*,2}$ follow the trail. This is a 48-bit filter on the pairs and thus we are left with 2^{42} candidate pairs. Then we guess the differences in bytes $B^1_{1,1}$ and $B^1_{3,3}$ (eight possibilities due to impossible input/output constraints in five

[3] For each byte difference there are only 8 possibilities that would not contradict with the five known differences in the 1st and 2nd rounds.

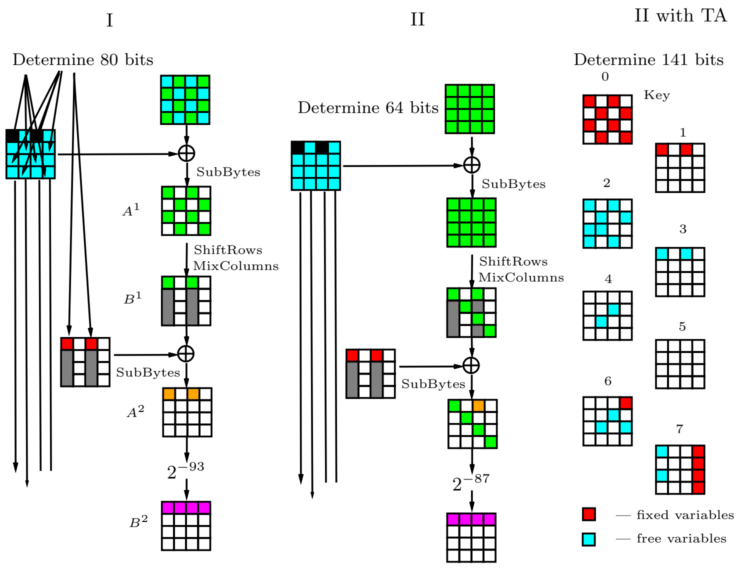


Fig. 4. Key recovery steps

corresponding S-boxes of the 1st and 2nd rounds), and undo the MixColumns.[4]. As a result, we have $2^{42+6} = 2^{48}$ key candidates from all pairs counting on a $64+16 = 80$ bit key (64 from K^0 and 16 from bytes $K^1_{1,1}, K^1_{3,3}$). No wrong key guesses survive this step. We find the remaining 12 bytes of K^1 by exhaustive search in 2^{96} steps. The total complexity of the attack is thus dominated by the $2^{96+35} = 2^{131}$ complexity of the first step.

Second step with TA. If we want to avoid the chosen-ciphertext framework, there is a way to combine the knowledge of 80 key bits on top and 35 key bits in the middle with a bit higher complexity. The problem is that the fixed bits are positioned far from one another (3 key schedule rounds apart), so it seems hard to make an efficient exhaustive search on the remaining part of the key.

We solve this problem by running the triangulation algorithm on the key schedule only, where 15 bytes are marked as fixed. The algorithm outputs 17 bytes in different subkeys as free variables. Then we assign these variables randomly, choose admissible values for the remaining ones, and thus define the key guess. There are $2^{17\cdot 8+5} = 2^{141}$ possible assignments, which would determine the complexity of the key recovery.

5 New Design Criteria for Block Ciphers

Our results imply new design criteria for the key schedules of the block ciphers. First of all, *local collisions* should be prevented. Although it is usually easy to

[4] We use the knowledge of the key byte $K^1_{0,0}$ to find the differences in the green diagonal at the 2nd round.

arrange a local collision in one round, there should be no good patterns for several rounds. A slow diffusion in the AES-256 key schedule helped us to concatenate many local collisions into a differential trail for the whole cipher. This should not be possible in a good cipher.

The key schedule should be also desynchronized with respect to the internal state. The active injection bytes in our AES trails are always located in the first row, which is not rotated. Therefore, the differences to be cancelled in a local collision should be located in the same column, or exactly 4 columns to the right in the subkey. This shift is preserved by the key schedule round, which should be certainly avoided.

Slow diffusion in the key schedule makes it also vulnerable to the triangulation algorithm or similar tools. Our preliminary analysis shows that the triangulation algorithm can cover up to two times the number of rounds needed for the full diffusion. If the key schedule in AES was ideal, we would be able to solve systems of equations on at most three-four rounds, while now we can attack five.

Finally, if one considers known or chosen key attacks as a threat he needs to add two extra full diffusions on top of the number of rounds secure against standard statistical attacks.

6 Conclusions

In this paper we show a chosen key distinguisher for the 256-bit key AES with almost practical complexity of $q \cdot 2^{67}$ queries and negligible memory. It was verified by computing partial q multicollisions in time $q \cdot 2^{37}$ which takes several hours on a PC. We also show the first related-key attack on the full AES-256 with 2^{96} data and time complexity and 2^{65} memory which works for 1 out of every 2^{35} keys on average.

Acknowledgements. The authors wish to thank anonymous reviewers for their valuable comments. We also wish to thank Christian Rechberger for useful discussions. Dmitry Khovratovich is supported by PRP "Security & Trust" grant of the University of Luxembourg. Ivica Nikolić is supported by the Fonds National de la Recherche Luxembourg grant TR-PHD-BFR07-031.

References

1. Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
2. Black, J.: The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 328–340. Springer, Heidelberg (2006)
3. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)

4. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
5. Daemen, J., Rijmen, V.: The Design of Rijndael. AES — the Advanced Encryption Standard. Springer, Heidelberg (2002)
6. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2000)
7. Jaulmes, É., Joux, A., Valette, F.: On the security of randomized cbc-mac beyond the birthday paradox limit: A new construction. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 237–251. Springer, Heidelberg (2002)
8. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
9. Khovratiovich, D., Biryukov, A., Nikolić, I.: Speeding up collision search for byte-oriented hash functions. In: CT-RSA 2009. LNCS, vol. 5473, pp. 164–181. Springer, Heidelberg (2009)
10. Kim, J., Hong, S., Preneel, B.: Related-key rectangle attacks on reduced AES-192 and AES-256. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 225–241. Springer, Heidelberg (2007)
11. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
12. National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard (November 2001), `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
13. National Security Agency (NSA). National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information (June 2003), `http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf`
14. Patarin, J.: A proof of security in $O(2^n)$ for the xor of two random permutations. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 232–248. Springer, Heidelberg (2008)
15. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multi-collisions. IEICE Transactions 91-A(1), 39–45 (2008)

A Details on Trails

The actual differences. The actual difference values should satisfy conditions imposed by the S-box properties, the presence of MC-columns in the subkeys, etc. We also require that differences in rounds 5 and 7–13 propagate through S-boxes with maximal probability — 2^{-6}. We programmed the search for the actual values and validated that those conditions do not lead to contradictions. The search has a negligible cost. We thus have defined the differences Δ_K (determined by subkeys K^0 and K^1) and Δ_C (determined by the subkey K^{14}). In the following subsections we list the actual differences in the trails.

Davies-Meyer trail (the first two rounds)

i	Plaintext		Subkey	Ciphertext			
0	01 01 01 01		0f 0e 0f 0e	01 01 01 01			
	00 00 00 00		07 07 07 07	00 00 00 00			
	00 00 00 00		07 07 07 07	00 00 00 00			
	00 00 00 00		09 09 09 09	00 00 00 00			
i	After SB	After MC	Subkey	i	After SB	After MC	Subkey
1	30 0f 44 b0	65 00 2b 00	37 00 37 00	2	1b 00 07 00	0c 00 0e 00	0f 01 0e 00
	7c b5 93 08	1f 2c 1f 00	1f 00 1f 00		00 12 00 00	07 00 07 00	07 00 07 00
	78 d6 c2 57	1f 00 e2 00	1f 00 1f 00		00 00 1a 00	07 00 07 00	07 00 07 00
	e7 c3 29 03	21 00 21 33	21 00 21 00		00 00 00 16	09 00 09 00	09 00 09 00

Related-key distinguisher (the first two rounds)

i	Plaintext		Subkey	Ciphertext			
0	0e 0e 0e 0e		0f 0e 0f 0e	01 01 01 01			
	07 07 07 07		07 07 07 07	00 00 00 00			
	07 07 07 07		07 07 07 07	00 00 00 00			
	09 09 09 09		09 09 09 09	00 00 00 00			
i	After SB	After MC	Subkey	i	After SB	After MC	Subkey
1	1f 00 1f 00	3e 00 3e 00	37 00 37 00	2	07 00 07 00	0e 00 0e 00	0f 01 0e 00
	00 00 00 00	1f 00 1f 00	1f 00 1f 00		00 00 00 00	07 00 07 00	07 00 07 00
	00 00 00 00	1f 00 1f 00	1f 00 1f 00		00 00 00 00	07 00 07 00	07 00 07 00
	00 00 00 00	21 00 21 00	21 00 21 00		00 00 00 00	09 00 09 00	09 00 09 00

Note on the colors. Differential trails are given in our figures in a truncated form, we marked distinct difference values with different colors. The reader does not need to care about the actual values in order to understand how the trail is constructed. However all the trail differences are provided in the tables of this Appendix.

The white cell stands for zero difference in a byte, the non-white cells stand for the non-zero differences. The same colors mean the same values except for the green, which denotes arbitrary differences. The exact relation between the colors and the values can be derived from the list of the actual differences. Grey and blue columns stand for MC-columns. In a spoiled MC-column one byte is marked with another color.

Differential trail for finding multicollisions:

Table 2. Multicollision trail

i	Plaintext		Subkey	Ciphertext			
0	00 00 00 00		0f 0e 0f 0e	01 01 01 01			
	00 00 00 00		07 07 07 07	00 00 00 00			
	00 00 00 00		07 07 07 07	00 00 00 00			
	00 00 00 00		09 09 09 09	00 00 00 00			
i	After SB	After MC	Subkey	i	After SB	After MC	Subkey
1	30 5c e1 b0	65 00 02 00	37 00 37 00	2	1b 00 07 00	0c 00 0e 00	0f 01 0e 00
	7c b5 ed 72	1f 25 1f 00	1f 00 1f 00		00 12 00 00	07 00 07 00	07 00 07 00
	a6 d6 c2 16	1f 00 e2 00	1f 00 1f 00		00 00 1a 00	07 00 07 00	07 00 07 00
	82 eb 29 03	21 00 21 33	21 00 21 00		00 00 00 16	09 00 09 00	09 00 09 00
3	1f 1f 00 00	3e 3e 00 00	37 37 00 00	4	07 07 00 00	0e 0e 00 00	0f 0e 00 00
	00 00 00 00	1f 1f 00 00	1f 1f 00 00		00 00 00 00	07 07 00 00	07 07 00 00
	00 00 00 00	1f 1f 00 00	1f 1f 00 00		00 00 00 00	07 07 00 00	07 07 00 00
	00 00 00 00	21 21 00 00	21 21 00 00		00 00 00 00	09 09 00 00	09 09 00 00
5	1f 00 00 00	3e 00 00 00	37 00 00 00	6	07 00 00 00	0e 00 00 00	0f 01 01 01
	00 00 00 00	1f 00 00 00	1f 00 00 00		00 00 00 00	07 00 00 00	07 00 00 00
	00 00 00 00	1f 00 00 00	1f 00 00 00		00 00 00 00	07 00 00 00	07 00 00 00
	00 00 00 00	21 00 00 00	21 00 00 00		00 00 00 00	09 00 00 00	09 00 00 00
7	1f 1f 1f 1f	3e 3e 3e 3e	3e 3e 3e 3e	8	00 00 00 00	00 00 00 00	01 00 01 00
	00 00 00 00	1f 1f 1f 1f	1f 1f 1f 1f		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	1f 1f 1f 1f	1f 1f 1f 1f		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	21 21 21 21	21 21 21 21		00 00 00 00	00 00 00 00	00 00 00 00
9	1f 00 1f 00	3e 00 3e 00	3e 00 3e 00	10	00 00 00 00	00 00 00 00	01 01 00 00
	00 00 00 00	1f 00 1f 00	1f 00 1f 00		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	1f 00 1f 00	1f 00 1f 00		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	21 00 21 00	21 00 21 00		00 00 00 00	00 00 00 00	00 00 00 00
11	1f 1f 00 00	3e 3e 00 00	3e 3e 00 00	12	00 00 00 00	00 00 00 00	01 00 00 00
	00 00 00 00	1f 1f 00 00	1f 1f 00 00		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	1f 1f 00 00	1f 1f 00 00		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	21 21 00 00	21 21 00 00		00 00 00 00	00 00 00 00	00 00 00 00
13	1f 00 00 00	3e 00 00 00	3e 00 00 00	14	00 00 00 00	00 00 00 00	01 01 01 01
	00 00 00 00	1f 00 00 00	1f 00 00 00		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	1f 00 00 00	1f 00 00 00		00 00 00 00	00 00 00 00	00 00 00 00
	00 00 00 00	21 00 00 00	21 00 00 00		00 00 00 00	00 00 00 00	00 00 00 00

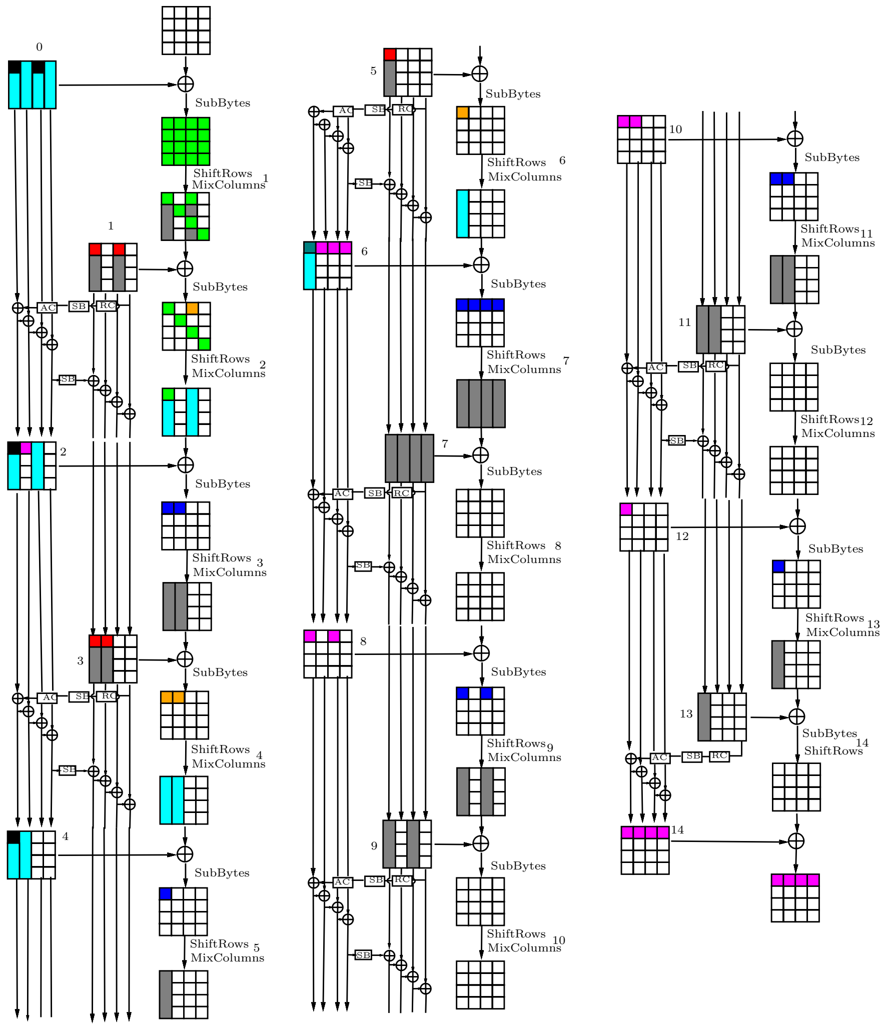


Fig. 5. Multicollision trail. The actual values are given in Table 2.

Cryptanalysis of C2

Julia Borghoff[⋆], Lars R. Knudsen, Gregor Leander, and Krystian Matusiewicz[⋆]

DTU Mathematics,
Technical University of Denmark
{J.Borghoff,Lars.R.Knudsen,G.Leander,K.Matusiewicz}@mat.dtu.dk

Abstract. We present several attacks on the block cipher C2, which is used for encrypting DVD Audio discs and Secure Digital cards. C2 has a 56 bit key and a secret 8 to 8 bit S-box. We show that if the attacker is allowed to choose the key, the S-box can be recovered in 2^{24} C2 encryptions. Attacking the 56 bit key for a known S-box can be done in complexity 2^{48}. Finally, a C2 implementation with a 8 to 8 bit secret S-box (equivalent to 2048 secret bits) and a 56 bit secret key can be attacked in $2^{53.5}$ C2 encryptions on average.

Keywords: block cipher, S-box recovery, key recovery, boomerang attack, C2, Cryptomeria.

1 Introduction

C2 is the short name for Cryptomeria, a proprietary block cipher defined and licensed by the 4C Entity (a consortium consisting of IBM, Intel, Matsushita and Toshiba) [3]. According to Wikipedia, "It (...) was designed for the CPRM/CPPM Digital Rights Management scheme which is used by DRM-restricted Secure Digital cards and DVD-Audio discs." [4]. 4C Entity has published a specification of C2 in [2].

C2 is a 10-round Feistel cipher with 64-bit blocks and a 56-bit key. The S-box is secret and available under license from the 4C Entity. Therefore, one might consider the S-box as part of the secret key.

A CPRM compliant device is given a set of secret device keys when manufactured. These keys are used to decrypt certain data of the media to be protected, in order to derive the media keys which have been used in the encryption of the main media data. The device keys can be revoked.

The specification of the system gives rise to several attack scenarios for C2.

1. The 56-bit key can be chosen by the attacker, who will attempt to determine the values in the secret S-box.
2. The S-box is known to the attacker, who will attempt to determine the value of a secret 56-bit key.
3. The 56-bit key and the S-box are unknown to the attacker, who will attempt to determine the values of both.

⋆ The author is supported by a grant from the Danish Research Council for Technology and Production Sciences grant number 274-07-0246.

In this paper we attack C2 in all three scenarios. The first attack requires 2^{24} chosen plaintext queries with negligible amount of other computations in the on-line phase. The complexity of the second attack is around 2^{48} of adaptive chosen ciphertext queries and a similar amount of computations. The third attack requires $2^{53.5}$ adaptive chosen ciphertext queries.

The first attack depends on the details of the key schedule. We show that by carefully selecting the value of the 56-bit key, we can ensure that only a limited number of three S-box entries are used in the first seven rounds of encryption of a chosen plaintext. By a trail-and-error approach these three entries can be determined. Subsequently other entries of the S-box can be determined in a similar approach. The attack has been successfully implemented and recovers the whole (secret) S-box in less than 30 seconds on a standard PC.

The second and third attacks make use of so-called boomerangs [12]. A study of the differential properties for C2 shows that there exist differential characteristics with good probabilities for up to 5 rounds of the total 10 rounds. These characteristics can be extended to more rounds but with a dramatic decrease in probability. It turns out that the differential characteristics can also be specified for 5 rounds the decryption operation of C2 with similar good probabilities. The average probability of the best such 5-round differential characteristics is 2^{-11}. The differential characteristics can be used to construct a boomerang, which has an average probability of 2^{-44}. One remarkable feature of this boomerang (and others) is that it exists regardless of what S-box is used.

We successfully generated plaintext pairs following the boomerang for various keys to verify the heuristic running times and to demonstrate the practical relevance of our attack.

It should be noted that, even though it has a better overall complexity, the second attack might still be slower in practice than a simple brute force attack which can of course be nicely distributed. Actually such a brute force attack on C2 has been carried out [1] (unsuccessfully as the S-box guess turned out to be wrong).

For the third attack scenario, brute force is clearly not an option, as not only the key but the entire S-box would have to be guessed, all together 2104 bits (or 1740 if we assume S-box is a permutation).

The only cryptanalytical result on C2 we are aware of is the S-box recovery attack (scenario 1) on 8 rounds of the cipher by Weinmann [13].

The rest of this paper is organized as follows. We start with a brief description of the cipher in Section 2. We present S-box recovery attack in Section 3. Then we discuss finding differential characteristics in Section 4. We present a key recovery attack for a known S-box in Section 5 followed by a key and S-box recovery attack in Section 6. Finally, we close with some conclusions.

2 Description of C2

In this section we fix our notation and present a description of the cipher.

2.1 Notation

Throughout the paper we will use the following notation.

- L_i, R_i – left and right word after $i = 1, 2, \ldots, 10$-th round of encryption (L_0, R_0 is the plaintext)
- $\mathsf{rotl}_m(b, n)$ – cyclic rotation of m bit sequence b by n positions left
- $X_{i,j}$ – j-th bit of word X_i
- $X_{i,p..q}$ – sequence of consecutive bits $X_{i,p}, X_{i,p+1}, \ldots, X_{i,q}$, e.g. $X_{i,0..7}$ is the least significant byte of X_i.
- $X \oplus Y$, $X \boxplus Y$ – respectively, bitwise XOR, addition modulo 2^{32} of words X and Y,

2.2 The Block Cipher C2

C2 [2] is a block cipher with 64-bit blocks and 56-bit keys. It consists of 10 Feistel rounds, each one using a 32-bit round key rk_i. The round function can be described as

$$\begin{aligned}
L_{i+1} &= R_i \\
X &= (R_i \boxplus rk_i) \oplus \texttt{0x2765ca00} \\
Z_{i,0..7} &= S[X_{i,0..7}] \\
Z_{i,8..15} &= X_{i,8..15} \oplus \mathsf{rotl}_8(Z_{i,0..7}, 1) \\
Z_{i,16..23} &= X_{i,16..23} \oplus \mathsf{rotl}_8(Z_{i,0..7}, 5) \\
Z_{i,24..31} &= X_{i,24..31} \oplus \mathsf{rotl}_8(Z_{i,0..7}, 2) \\
R_{i+1} &= L_i \boxplus (Z_i \oplus \mathsf{rotl}_{32}(Z_i, 9) \oplus \mathsf{rotl}_{32}(Z_i, 22)), \quad i = 0, \ldots, 9
\end{aligned}$$

and is illustrated in Fig. 1. We denote by Ψ the $GF(2)$-linear function that maps bits of Y_i to the bits of U_i, this part is framed in the dotted box in the figure and the explicit equations are given in Section A.1 in the Appendix.

Note that the original reference code [2] describes it slightly differently using three byte constants, but we present here a simpler, equivalent form using only one constant $C = \texttt{0x2765ca00}$.

The key schedule produces 10 round keys $rk_0, \ldots, rk_9$ out of 56-bit master key K in the following way.

$$\begin{aligned}
K'_i &= \mathsf{rotl}_{56}(K, 17 \cdot i) \ , \\
rk_i &= K'_{i,0..31} \boxplus (S[K'_{i,32..39} \oplus i] \ll 4), \quad i = 0, \ldots, 9 \ .
\end{aligned}$$

The exact numbers of bits of the master key used in each round are also given in Table 2 in the Appendix for reference.

Both the round transformation and the key scheduling use an 8-bit secret S-box S. An example S-box provided by 4C for the purpose of validating the implementations is available online [5].

3 Recovering Secret S-box with Chosen Key Attack

Our attack to recover the S-box when we are allowed to choose a encryption key is based on the observation that some keys generate only very few different inputs

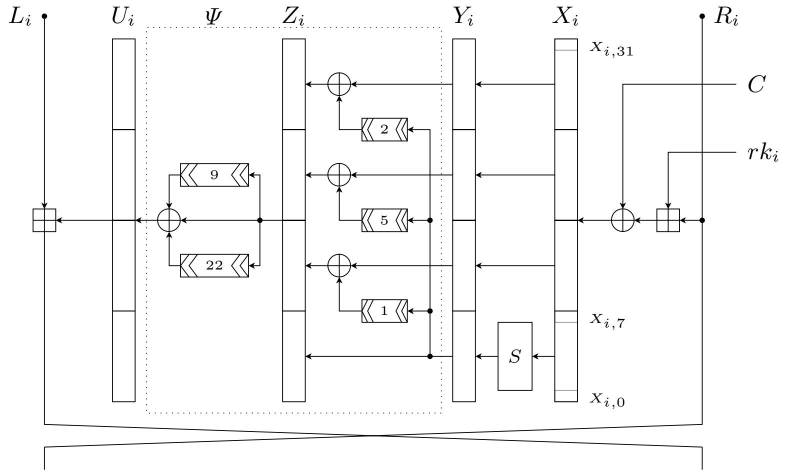


Fig. 1. Equivalent description of the round transformation of C2

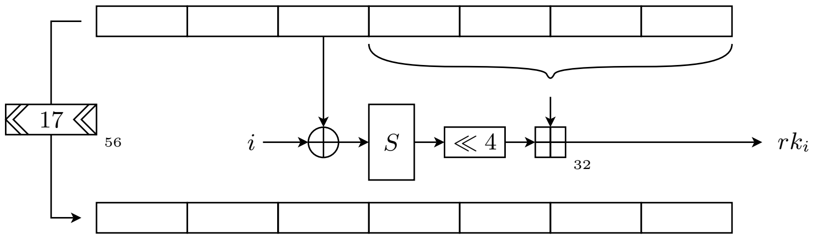


Fig. 2. One step of the key scheduling algorithm generates 32-bit round key rk_i

to the secret S-box in the key scheduling. It is easy to verify using a computer search that the smallest number of inputs generated in the key scheduling is three. An example of such a master key is

`0x40, 0x84, 0x88, 0x40, 0x02, 0x80, 0x09`

and the inputs generate to the S-box in rounds 1 to 10 are the following

`0x88, 0x4, 0x27, 0x27, 0x4, 0x4, 0x27, 0x27, 0x88, 0x88`

For the attack we first fix the above key and guess the possible outputs of the S-box for the inputs `0x04`, `0x27` and `0x88`. For each possible guess we generate one plaintext that, under the assumption that our guess is correct, does not trigger any additional entries in the secret S-box for 7 rounds. For such a plaintext, again under the assumption that our guess is correct, we know the output of the encryption process after 7 rounds, i.e. (L_7, R_7). As explained below, generating a plaintext for one fixed key guess requires approximately $2^{19.25}$ C2-encryptions and as there are 2^{24} possible values for the three entries in the secret S-box the complexity for this step is approximately $2^{43.25}$ C2-encryptions. However, this computation is independent of the actual S-box being attacked and therefore has to be done only once and is trivially parallelizable. We computed a table containing one plaintext for each guess. The actual running time was 96 hours and the size of the table is less than 400 MByte. The details are in Section 3.1.

When attacking an actual device or implementation using a secret S-box we proceed as follows. We encrypt each plaintext in the table –corresponding to one possible guess of the three S-box entries– using the device and observe the ciphertext. If our guess is correct we know the output after round 7. As explained in Section 3.2 it is possible to check if the observed ciphertext fits to our guess of the 7th round output. This test will never fail for the right guess and has a (heuristic) probability of accepting a wrong guess with a probability of 2^{-29}. Thus, on average, only the right guess will survive. Using the outlined approach we can recover three S-box entries with 2^{24} encryptions using the actual device and marginal overhead for the test.

After the first three entries have been recovered we continue in a very similar way. First, it is now easy to recover (up to) three additional entries corresponding to the inputs triggered in the last three rounds without querying the device. For all other entries we now generate plaintexts that do not trigger any unknown inputs in the first six rounds. Using the three round test explained in Section 3.2 on any possible output of the S-box in round 7 we can recover the output of the S-box in the 7th round and later recover the output of the S-box in the last three rounds again. Assuming that the inputs to the S-box in rounds $7, 8, 9$ and 10 behave randomly an estimate for the complexity (in terms of C2 encryptions) of successfully recovering the whole S-box is derived from the well known coupon collector's problem [8, Section II.7] and given by

$$C\frac{(256 \cdot H_{256})}{4} \approx 2^{19.4},$$

where H_n is the nth harmonic number and C is the complexity to generate a plaintext that fit for 6 rounds. As explained in Section 3.1, C can be upper bounded by

$$C \leq \left(\frac{256}{6}\right)^2.$$

However, it turns out that those inputs do not behave purely random and experimentally we measured a slightly higher complexity of $2^{20.2}$ as an average of 10000 tries (100 tests for 100 randomly generated S-boxes). Summarizing, when we are allowed to choose an encryption key, the S-box can be recovered with less than 2^{24} queries to the device on average. Of course, the actual running time highly depends on the encryption speed of the device, but for an implementation on a standard PC the whole S-box can be recovered in less than 30 seconds.

3.1 Generating Plaintexts That Fit for Seven Rounds

We next describe a procedure to generate a plaintext such that for known (or guessed) round keys and a set of known (or guessed) input/output pairs S the inputs to the Sbox in the first seven rounds are within this set S. First note that a naive method would be to randomly generate plaintexts and verify if the plaintext fulfills the conditions in all seven rounds. Under the assumption that those inputs behave randomly, the effort to generate such a plaintext is $(\frac{256}{|S|})^7$. For the first part of the attack, where $|S| = 3$, this is approximately $2^{44.9}$. As we have to generate not only one, but 2^{24} such plaintext the complexity of this naive approach is too high. However, it is easy to generate plaintexts that fulfil the conditions for four out of the seven rounds by construction. Then, again assuming things behave randomly, the effort is reduced to $(\frac{256}{|S|})^3$ which for $|S| = 3$ and 2^{24} plaintexts to be generated gives an overall complexity of approximately $2^{43.25}$.

Note that in the following the names of variables refers to Figure 1. To get the inputs in round 2 up to round 5 correct we first choose those inputs, i.e. we fix $X_{1,0..7}, X_{2,0..7}, X_{3,0..7}$ and $X_{4,0..7}$ to arbitrary inputs in the set S. Furthermore we choose $X_{2,8..31}$ and $X_{3,8..31}$ randomly. With this we can compute

$$\begin{aligned}
R_{1,0..7} &= (X_{1,0..7} \oplus C_{0..7}) - rk_{1,0..7} \pmod{2^8} \\
R_{2,0..7} &= (X_{2,0..7} \oplus C_{0..7}) - rk_{2,0..7} \pmod{2^8} \\
R_{3,0..7} &= (X_{3,0..7} \oplus C_{0..7}) - rk_{3,0..7} \pmod{2^8} \\
R_{4,0..7} &= (X_{4,0..7} \oplus C_{0..7}) - rk_{4,0..7} \pmod{2^8}.
\end{aligned}$$

Next, observe that for any 8 bit vector x it holds that

$$F(X \oplus (x << 23))_{0..7} = F(X)_{0..7} \oplus x$$

where F denotes the function mapping X_i to U_i. In particular we can choose bytes x and y such that

$$F(X_2 \oplus (x << 23))_{0..7} - R_{3,0..7} = R_{1,0..7} \pmod{2^8}$$

and

$$F(X_3 \oplus (y << 23))_{0..7} + R_{2,0..7} = R_{4,0..7} \pmod{2^8}.$$

Thus, if we choose

$$L'_3 = (X_2 \oplus (x << 23) \oplus C) - rk_2$$

and

$$R'_3 = (X_3 \oplus (y << 23) \oplus C) - rk_3$$

and decrypt this for three rounds to get a plaintext (L'_0, R'_0) we ensured that for this plaintext from the second until the fifth round all inputs to the Sbox are as previously fixed – and thus in the set S. We experimentally verified that the complexity of generating plaintext that also fit in the first, sixth and seventh round for $|S| = 3$ is approximately $(\frac{256}{3})^3 \approx 2^{19.25}$ as predicted by the heuristic. The overall running time to generate all 2^{24} plaintexts for each guess was distributed to 100 CPUs and took less than one hour.

3.2 A Three Round Test

To make sure we guessed the S-box entries right we need to check if the output of the 7th round based on the guess and the ciphertext match, i.e. encrypting (L_7, R_7) with three rounds gives us the right ciphertext (L_{10}, R_{10}). This test would be trivial if we knew the S-box. However, we still can do it efficiently and with very good probability even without knowing the S-box.

Since we know the values of R_7 and R_{10}, we can compute $U_8 = R_{10} - R_7$ and going backwards through the F-function, we can determine Y_8. Since we do not know the S-box, we know only 24 msb bits of X_8. We have guessed the round key rk_8 and this means only if we knew whether a carry in the modular addition occurred or not, we would know 24 msb bits of R_8 and L_9. Now, using this knowledge and the values of L_7 and L_{10} we can determine 24 msb bits of $U_7 = R_8 - L_7$ and $U_9 = L_{10} - L_9$. Again, we do not know the carry bit so we have to test two possibilities for each of the words, either assuming a carry occurred or not. Provided that the carries are as predicted, we know exactly 24 msb bits of U_7 and U_9. Let us focus on the 7th round first. To test whether the input and the ciphertext match, we want to compare the values of U_7 obtained by the above procedure with $U'_7 = \Psi(Y_7)$, where Ψ is a $GF(2)$-linear map (marked with dotted box in Fig. 1). We cannot compare U_7 with U'_7 directly because we do not know bits $U_{7,0..7}$ and the unknown output of the S-box masks bits of U'_7. However, we can compare linear combinations of bits of U_7 and $\Psi(Y_7)$ that do not depend on any of the unknown bits $U_{7,0..7}$ and $Y_{7,0..7}$. There are 16 linear equations $\xi_j(U_7) = \xi_j(\Psi(Y_7))$ involving bits of U_7 and Y_7 that do not use any unknown bits. If the pair $(L_7, R_7), (L_{10}, R_{10})$ matches and we guessed all the carries correctly, all these equations will be satisfied. For an unrelated pair of inputs and outputs, this happens with probability 2^{-16}. The same happens for the test in round 10. We combine those two tests with a simple guessing of all the carries we need to know to obtain our testing procedure. For each of the two possible values of the carry in round 9, we test independently two possible carries in round 7 and round 10. If for any combination of these all the 32 pairs of check equations agree, we conclude the pair matches. Otherwise, we reject the pair.

This procedure always accepts right pairs $(L_7, R_7), (L_{10}, R_{10})$ as they will always produce a match in one of the tested carry combinations. To accept a wrong pair which is not coming from the encryption, all the 32 pairs of check equations would need to agree for one of the 2^3 combinations of carries. This happens with probability 2^{-29} if values are uniformly distributed. We experimentally verified that the probability is indeed around 2^{-29}. This is sufficient for us since we need to test only 2^{24} possibilities.

4 Search for S-box Independent Characteristics

The only components of C2 that are not linear over GF(2) are the S-box and the two modular additions. As the S-box is secret and therefore its differential behavior is unknown, we focus on characteristics not involving the S-box. Note that if the input to the round function has zero difference in the least significant byte $R_{i,0..7}$, this zero difference cannot be destroyed by carries in the modular key addition. Thus, we can search for characteristics independent of the S-box by focusing on characteristics with $R_{i,0..7} \oplus R'_{i,0..7} = 0$.

To search for these characteristics we consider a linear model of the round function, that is, we replace the modular addition by XORs and assume that the S-box is the identity (or any linear mapping as the characteristic will be independent of this choice anyway). This linear model of the round function

$$(L_i,\ R_i) = (R_{i-1},\ L_{i-1} \oplus F(R_{i-1}, K_i))$$

can be written as $(L_i, R_i) = (L_{i-1}, R_{i-1}) \cdot M$ where M is a 64×64 matrix over $GF(2)$. Furthermore, the condition that the input difference to the S-box, i.e., the least significant byte of the output difference, shall be zero can be described as $((L, R)M)Q = 0$ where Q corresponds to the projection on the least significant 8 bits. Thus, for the linearized version of the cipher, the problem of finding a characteristic which has a zero input differences to the S-box is reduced to the problem of calculating the kernel of the linear mapping $x \to x \cdot M \cdot Q$. The kernel of the matrix $K = [Q|M \cdot Q| \cdots |M^i \cdot Q]$ contains all differences which have a zero input difference to the S-box over $i+1$ rounds. This kernel is non trivial for $i \leq 8$ implying that for version of C2 where the modular additions are replaced by XORs a characteristic over 9 rounds with probability 1 exists *independently of the S-box.*

As modular additions are not linear over $GF(2)$ we need to estimate the probability that the modular addition behaves like an XOR. Here we are interested in the two following cases.

1. The probability that the key addition behaves like an XOR

$$\Pr[(C \boxplus K) \oplus ((C \oplus \alpha) \boxplus K) = \alpha]$$

where C and K are random bit strings and α is the known difference.

2. The probability that the addition of the left half and the output of function F behaves like an XOR

$$\Pr[(L \boxplus F) \oplus ((L \oplus \alpha) \boxplus (F \oplus \beta)) = \alpha \oplus \beta]$$

where L and F random bit strings and α and β fixed known differences.

These probabilities have been studied for example in [10] where it was shown that

$$\Pr[(C \boxplus K) \oplus ((C \oplus \alpha) \boxplus K) = \alpha] = 2^{-(\mathrm{hw}(\alpha) - \mathrm{msb}(\alpha))}$$

and

$$\Pr[(L \boxplus F) \oplus ((L \oplus \alpha) \boxplus (F \oplus \beta)) = \alpha \oplus \beta] = 2^{-(\mathrm{hw}(\alpha \vee \beta) - \mathrm{msb}(\alpha \vee \beta))}$$

where $\mathrm{hw}(\alpha)$ denotes the Hamming weight of α and $\mathrm{msb}(\alpha)$ the most significant bit.

Since the probability of an XOR-characteristic depends mainly on the Hamming weight of all the intermediate input differences, we searched for characteristics minimizing it. This problem is equivalent to searching for low weight code words in the linear code generated by the matrix $B \cdot [I|M|\cdots|M^i]$ where B is the basis matrix of the kernel of K. Such an approach has been used before for finding differential characteristics in dedicated hash functions, cf. [11].

The best five round characteristic we found is

$$\Delta = (00020800\,80200100)_x \rightarrow (80200100\,00020800)_x$$

which does not require non-zero differences to the S-box in any round, and it has Hamming weight 15 over all intermediate input differences. Using the above formulas from [10] one gets a probability of 2^{-12} for independent round inputs and keys. Experimentally, the probability for randomly chosen master keys and S-boxes was even better, namely approximately $2^{-11.17}$, which is due to a differential effect which takes place inside the 5-round characteristic.

The differential characteristic can be specified also for the last five rounds of C2 and the average probability was estimated to be similar to the one for the first five rounds.

5 Key Recovery Attack for a Known S-box

The five round characteristics described in Section 4 can be used to mount a boomerang attack on the whole cipher [12,9,6]. A boomerang attack is chosen plaintext and chosen ciphertext attack that involves four encryptions. We experimentally estimated (by testing 1000 random keys and multiplying probabilities of passing the first five and the last five rounds) that boomerangs exist with an average probability of $2^{-44.5}$. We observed that for all such boomerangs the pairs of texts followed the characteristic in the first round every time, but not always in later rounds. The reason one can obtain a boomerang anyway is the differential effect which is utilized also in the so-called rectangles [6]. We further observed a large variability in the probabilities over the keys and some keys were found for which the probability of the boomerang is as high as 2^{-32} but also there are keys for which no boomerangs were found. We present some of actual boomerangs we found in Table 1.

The possibility of finding boomerangs enables us to test if the differences in the first round propagate according to the characteristics. If not, we do not expect to get any boomerangs. We will use this observation to recover many bits of the first round key by a careful analysis of the carries appearing in the addition $R_0 \boxplus rk_0$. This method resembles the approach used by Contini and Yin to partially recover HMAC keys using a pseudo-collision differential for MD5 [7].

5.1 Recovering Bits of the First Round Key

Here we are going to describe how to recover up to 22 bits of the first round key by applying the boomerang attack outlined above.

Table 1. Examples of boomerang plaintext pairs for different keys and S-boxes

S-box used	key (hex)	plaintext
AES	00 00 00 00 00 00 00	5707aec0 48a9c942
	00 30 20 08 00 20 28	0f42cd03 b7b5f077
	'c' 'r' 'y' 'p' 't' '0' '9'	b4b32db5 589913dc
C2 facsimile [5]	00 00 00 00 00 00 00	3af32bac 960693e1
	ee 9b 7f 2b 7c 26 cd	69676fdc 339879d4
	'c' 'r' 'y' 'p' 't' '0' '9'	d6b44956 36771c9d

Given a plaintext pair (L_0, R_1) and $(L_0 \oplus \alpha, R_0 \oplus \beta)$ from a boomerang, we know (with overwhelming probability) the difference after the first round and also know that the difference of the right halves after the modular key addition is still β. Therefore the first round key has to fulfill the equation

$$(R_0 \boxplus rk_0) \oplus ((R_0 \oplus \beta) \boxplus rk_0) = \beta. \tag{1}$$

We denote the vector of carry bits of the modular addition of R_0 and rk_0 by $c(R_0, rk_0)$, i.e.

$$R_0 \boxplus rk_0 = R_0 \oplus rk_0 \oplus c(R_0, rk_0)$$

where then

$$c_{-1} = 0 \text{ and } c_i = R_{0,i} rk_{0,i} \oplus c_{i-1} R_{0,i} \oplus c_{i-1} rk_{0,i}.$$

Using this, (1) can be rewritten as

$$R_0 \oplus rk_0 \oplus c(R_0, rk_0) \oplus R_0 \oplus \beta \oplus rk_0 \oplus c(R_0 \oplus \beta, rk_0) = \beta$$

which is equivalent to

$$c(R_0, rk_0) = c(R_0 \oplus \beta, rk_0) \tag{2}$$

and furthermore implies

$$\beta_i \; rk_{0,i} = \beta_i \; c_{i-1}.$$

Thus, whenever $\beta_i = 1$ the previous carry bit –which potentially depends on all previous key bits – equals the key bit. On the downside, Equation 2 implies that we cannot extract any key bits beyond the most significant non-zero bit of β. Using the 5 round characteristic from Section 4 we can therefore at most recover 22 bits of the first round key using (2).

In the following we describe how bits of the round key can be found one at a time. Instead of using randomly chosen plaintexts (L_0, R_0) we start by fixing the 8 least significant bits of R_0 to zero. This ensures that $c_7 = 0$. Equation (2) implies that boomerangs with this additional constraint exist iff $rk_{0,8} = 0$.

Thus, if after sufficiently many tries, we do not find any boomerang, we can conclude that $rk_{0,8} = 1$. Let us estimate the probability of making a mistake there and wrongly assuming that $rk_{0,8} = 1$ while in reality it holds that $rk_{0,8} = 0$.

If 2^{-b} is the probability of a boomerang and we make our decision after $t2^b$ tries, then the error probability can be approximated by

$$(1 - 2^{-b})^{t2^b} = \left((1 - 2^{-b})^{2^b}\right)^t \approx \left(\frac{1}{e}\right)^t .$$

After recovering $rk_{0,8}$ we modify our choice of plaintexts adaptively depending on the recovered bit $rk_{0,8}$.

First, consider the case $rk_{0,8} = 0$. Here we generate plaintext pairs where the least significant 8 bits of R_0 equal 01000000. In this case $c_7 = 0$ if and only if $rk_{0,7} = 0$ or, equivalently, boomerangs exist only when $rk_{0,7} = 0$. Thus, after sufficiently many tries, we can with a good probability recover $rk_{0,7}$.

Next, consider the case where $rk_{0,8} = 1$. Here we again fix the least significant 8 bits of R_0 to 01000000 and again $c_7 = 0$ if and only if $rk_{0,7} = 0$. However, in this case boomerangs exist only when $rk_{0,7} = 1$.

This procedure can now be applied recursively to finally recover all the key bits $rk_{0,0...7}$. After those bits have been successfully recovered a very similar argument allows to recover the key bits $rk_{0,21...8}$.

Assuming the average complexity for finding the boomerang is 2^{44}, the overall complexity of this procedure to recover B bits for a random key can be estimated to

$$B \cdot \left(\frac{t2^{44} + 2^{44}}{2}\right) \tag{3}$$

and the error probability is approximately

$$1 - \left(1 - \left(\frac{1}{e}\right)^t\right)^B . \tag{4}$$

If we want to recover 8 bits with a success probability of more than 0.5 we have to choose $t = 2.48$ and the effort will be $2^{47.8}$. The remaining 48 bits of the master key can then be recovered with a brute force search.

If we want to recover all 22 bits with a success probability of more than 0.99 we have to choose $t = 7.7$ and the effort will be $2^{50.59}$.

Note that for a given key it is unclear at first what the probability for the boomerang actually is. However, there are several ways to deal with this problem. On possibility is to first get an estimate of the probability by running the boomerang search for randomly selected plaintexts. Another possibility is to double the time until we decide on a key bit when no boomerang has been found step by step until the right key has been found.

6 Key and S-box Recovery with Chosen Ciphertext Attack

The attack recovering the key and the S-box is again based on the boomerang attack outlined above. As explained in Section 5.1 we can recover the least significant 22 bits of the first round key with an average complexity of $2^{50.59}$ and an error probability less then 0.01. But turning the boomerang upside down, we can similarly recover 22 bits of the last round key with the same complexity. As explained in Section 6.1 it is possible to recover the remaining bits of these round keys and one entry of the secret S-box with an average complexity of 2^{52}. Thus with an effort of approximately 2^{53} we can recover the first and the last round key. This knowledge allows us to recover the second round key (see Section 6.2) with an average effort of $2^{45.32}$. The first two and the last round key together determine the entire master key uniquely (cf. Table 2 in the appendix). We are now in the position where we can recover additional entries of the secret S-box with an effort of 2^{44} by again applying the approach of Section 6.2. After recovering four more entries (with an effort of 2^{44+2}) of the S-box corresponding to what is triggered in the key scheduling in rounds $3, 4, 5$ and 6 we can use an attack very similar to the attack described in Section 3 to recover the remaining entries of the S-box. Namely, we guess the remaining three S-box entries triggered in the key scheduling in rounds $7, 8$ and 9. For each possible guess we generate a plaintext that does not trigger any unknown (or un-guessed) S-box entries in the first seven rounds. As we know or guessed 10 entries already the effort of generating such a plaintext is $(256/10)^3 \approx 2^{14}$. We encrypt each of those plaintexts and use the check of Section 3.2 to verify our guess. This way we will recover all 10 S-box entries used in the key scheduling and afterwards the remaining entries are recovered just as in Section 3 with a complexity less than 2^{20}. The complexity of recovering the S-box is therefore $2^{24+14} = 2^{38}$ and the overall complexity of the attack is

$$2 \cdot 2^{50.59} + 2 \cdot 2^{52} + 2^{45.3} + 2^{44+2} + 2^{38} + 2^{20} \approx 2^{53.5}$$

on average.

6.1 Recovering Remaining Unknown Round Key Bits

Once we know bits $rk_{0,0..21}$ of the first round key we can recover the remaining most significant bits of the round key and the output of the S-box using the carry behaviour of the left addition $L_0 \boxplus U_0$.

If we have a boomerang plaintext, we know that the following equation is true

$$(L_0 \boxplus U_0) \oplus [(L_0 \oplus \alpha) \boxplus (U_0 \oplus \Psi(\beta))] = \texttt{0x80000000} \ ,$$

where $\alpha = \texttt{0x00020800}$, Ψ is a linear function mapping bits of Y_0 to U_0 and we have $\Psi(\beta) = \texttt{0x80020800}$. Since the difference in the most significant bit always propagates linearly as it does not induce any carries, we can focus on a simplified version of the above equation

$$(L_0 \boxplus U_0) \oplus [(L_0 \oplus \alpha) \boxplus (U_0 \oplus \alpha)] = 0 \ .$$

Using the same method as in Section 5.1 we get

$$c(L_0, U_0) = c(L_0 \oplus \alpha, U_0 \oplus \alpha)$$

and it simplifies to the condition

$$\alpha_i(L_{0,i} \oplus U_{0,i} \oplus 1) = 0 \ . \tag{5}$$

Since α has bits 11 and 17 set, (5) allows us to determine bits $U_{0,11}$, $U_{0,17}$ by trying to find boomerang plaintexts for all of the four possible combinations of $L_{0,11}$, $L_{0,17}$ in parallel. One of the choices will yield a boomerang and it contains the right combination of values of $L_{0,11}$, $L_{0,17}$ that determine the values of bits of U_0.

We have $U_{0,11} = Y_{0,0} \oplus Y_{0,11} \oplus Y_{0,21}$ and we can compute the values of $Y_{0,11}, Y_{0,21}$ because we know R_0 and the round key bits $rk_{0,0..21}$. Thus, we learn one bit of the output of the S-box ($Y_{0,0}$). Furthermore, we get another equation $U_{0,17} = Y_{0,1} \oplus Y_{0,4} \oplus Y_{0,7} \oplus Y_{0,8} \oplus Y_{0,17} \oplus Y_{0,27}$. The complete system of equations describing bits of U_0 can be found in Section A.1 of the appendix.

The same principle can be used to recover more bits. In order to do this, we need differences to appear at other bit positions in the addition $L_0 \boxplus U_0$. We can achieve this by inducing carry chains in the first addition $R_0 \boxplus rk_0$ by appropriately setting some bits of the plaintext so that the difference $\beta =$ 0x80200100 will trigger more bit flips in $R_0 \boxplus rk_0$. More precisely, we find plaintexts R_0 such that

$$(R_0 \boxplus rk_0) \oplus [(R_0 \oplus \beta) \boxplus rk_0] = \beta \oplus \gamma \ .$$

for some carry-induced difference γ. Remember that Ψ mapping Y_0 to U_0 is linear and so this induces an extra difference $\Psi(\gamma)$, so $U_0 \oplus U_0' = \Psi(\beta) \oplus \Psi(\gamma)$.

Later, we try to compensate for this extra difference in U_0 by the additional difference $\Psi(\gamma)$ in L_0. This situation can be described as

$$(L_0 \boxplus U_0) \oplus [(L_0 \oplus \alpha \oplus \Psi(\gamma)) \boxplus (U_0 \oplus \Psi(\beta) \oplus \Psi(\gamma))] = \texttt{0x80000000}$$

If this equation holds (and we know this when we find a boomerang) we have the following conditions

$$(\alpha_i \oplus \Psi_i(\gamma))(L_{0,i} \oplus U_{0,i} \oplus 1) = 0$$

which allow us to determine bits of U_0 at positions i where $\alpha_i \oplus \Psi_i(\gamma) = 1$.

Because of the effect of Ψ, each bit in γ usually requires 3 additional compensating bits of the difference in L_0 and this means we need to search for 8 boomerangs in parallel to determine the right values of L_0. After we find one, we obtain three more equations as explained before.

The complexity of this procedure depends on the configuration of carry chains we are able to induce and this depends on the round key. Assuming we can extend the difference in $\beta =$ 0x80200100 at position 8 to chains at positions 8-9, 8-10, 8-11, 8-12, 8-13, 8-14, 8-15 (so γ is 00000200, 00000600, 00000c00, etc.) we get enough equations to uniquely determine the unknown bits of Y_0. We need to test 2^3 combinations of values of bits in L_0 and the total complexity is $2^3 \cdot 2^3 \cdot 2^{44} = 2^{52}$ where 2^{44} is the cost of finding the boomerang plaintext. For other configurations of secret key bits we may not be able to extend γ by one bit at a time and we will need to test more bits in L_0 each time. In that situation we usually need to test less cases though because we learn more bits of U_0 at the same time. The exact increase in complexity very much depends on a particular case.

Note that we can always perform a search for the 13 missing bits by randomly choosing plaintext pairs (L_1, R_1) and (L_1', R_1') with a difference corresponding to the second round difference of our 5 round characteristic, decrypting them using all possible guesses for the missing 13 bits and searching (in parallel) a boomerang for all 2^{13} pairs. This upper bounds the complexity of recovering the remaining bits in the first round by $2^{13} \cdot 2^{44} = 2^{57}$.

A possible speed-up is to use both ends of the boomerang – if we find a boomerang plaintext we have actually two plaintexts that follow the characteristics in the first round of encryption. This can reduce the necessary number of boomerangs we need to find to completely recover the round key and the output of the S-box.

6.2 Attacking the Second Round

Knowing the entire first round key and one entry of the secret S-box we can (provided we fix $R_{0,0..7} = L_{1,0..7}$ to keep the input to the S-box the same) start the boomerang in the second round. For this we choose pairs (L_1, R_1) and (L_1', R_1') with the input difference of the best five round characteristic and compute backwards the corresponding values for (L_0, R_0) and (L_0', R_0'). For the lower part of the boomerang we can now use our 5 round characteristic truncated to the first 4 rounds. This shortened boomerang will give pairs (L_0'', R_0'') and (L_0''', R_0''') with an average probability of $2^{-(2\cdot 11+2\cdot 8)} = 2^{-38}$. We cannot directly compare the corresponding pairs (L_1'', R_1'') and (L_1''', R_1''') as with high probability we do not know the S-box entry to decrypt in the first round. However, we can still check that the right half difference $R_0'' \oplus R_0'''$ is $\beta = \texttt{0x80200100}$ as desired. Furthermore, by exhaustively trying all possible output values for the S-box for pairs with the correct right half difference, we get an additional $32 - 8$ bit check for the left half difference. Thus, with high probability we detect correctly pairs following the boomerang characteristic.

Now, repeating the procedures outlined in Section 5.1 and 6.1 we first recover the 22 least significant bits of the second round key $(rk_{1,0..21})$ and afterwards the remaining 7 bits of the round key $(rk_{1,22..29})$ as well as one additional entry of the S-box. Note that the bits $rk_{1,29..31}$ are known from the last round key. The complexity of this is now

$$22 \cdot \left(\frac{7.7 \cdot 2^{38} + 2^{38}}{2} \right) \approx 2^{44.58}$$

for the first step and $2^3 2^3 2^{38} = 2^{44}$ for the second step.

Using this shortened boomerang described in the last section, we can moreover recover arbitrary S-box entries by fixing $R_{1,0..7}$ appropriately. The complexity for this is again 2^{44} on average.

7 Conclusions

We have shown three kinds of attacks on the block cipher C2.

When we are allowed to set the encryption key once and then encrypt plaintexts chosen by us, we can recover the secret S-box with only 2^{24} queries to the device and a reasonable precomputation phase that we have already done. The attack implemented on a PC recovers the whole S-box in less than 30 sec. Due to a low query complexity, we believe that this attack could be applied in practice to recover S-box from an actual device.

When the S-box is known, we present a boomerang attack that recovers the key with complexity equivalent to 2^{48} C2 encryptions and works for all possible S-boxes.

For the most difficult case, when both the key and the S-box are unknown and we are faced with an equivalent of at least 1740-bit long key, we present an attack that recovers both of them with complexity of around $2^{53.5}$ queries to the encryption device.

Furthermore, we show that the main strength of the cipher lies in the modular additions rather than the S-box. With modular additions replaced by XORs, one can find 9 round differentials with probability 1 and boomerangs for all 10 rounds with probability 1, both regardless of the S-box is used.

All our attacks do not assume anything about the S-box, not even its bijectiveness. Moreover, the first attack does not depend the choice of the linear mixing map Ψ used in the round function.

It is surprising that the addition of the secret S-box does not substantially improve the overall security of the design. It shows that to achieve the desired effect, the algorithm using a secret S-box must be designed very carefully. Probably a better option would be to use a longer secret key instead.

References

1. Distributed C2 brute force attack : Status page, `http://www.marumo.ne.jp/c2/bf/status.html` (accessed on 12/02/2009)
2. C2 Block Cipher Specification, Revision 1.0 (2003), `http://www.4Centity.com`, used to be available online from 4C Entity, `http://edipermadi.files.wordpress.com/2008/08/cryptomeria-c2-spec.pdf`
3. 4C Entity. Wikipedia article, `http://en.wikipedia.org/wiki/4C_Entity` (accessed on 11/02/2009)
4. Cryptomeria cipher. Wikipedia article, `http://en.wikipedia.org/wiki/Cryptomeria_cipher` (accessed on 11/02/2009)
5. 4C Entity. C2 facsimile s-box, `http://www.4centity.com/docs/C2_Facsimile_S-Box.txt`
6. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
7. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)

8. Feller, W.: An introduction to probability theory and its applications, 3rd edn., vol. I. Wiley, Chichester (1968)
9. Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (2001)
10. Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002)
11. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of step-reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
12. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
13. Weinmann, R.-P.: Algebraic S-Box recovery: the case of Cryptomeria. Presentation at Echternach Seminar on Symmetric Cryptography, Echternach, Luxembourg (11/01/2008), `http://wiki.uni.lu/esc/docs/rpw_friday_algebraic_sbox_recovery.pdf`

A Appendix

A.1 Equations Describing $\Psi : Y \to U$

$$\begin{aligned}
u_0 &= y_0 + y_1 + y_2 + y_{10} + y_{23} & u_{16} &= y_0 + y_3 + y_7 + y_{16} + y_{26}\\
u_1 &= y_1 + y_2 + y_6 + y_{11} + y_{24} & u_{17} &= y_1 + y_4 + y_7 + y_8 + y_{17} + y_{27}\\
u_2 &= y_2 + y_3 + y_7 + y_{12} + y_{25} & u_{18} &= y_0 + y_2 + y_5 + y_9 + y_{18} + y_{28}\\
u_3 &= y_0 + y_3 + y_4 + y_{13} + y_{26} & u_{19} &= y_1 + y_3 + y_6 + y_{10} + y_{19} + y_{29}\\
u_4 &= y_1 + y_4 + y_5 + y_{14} + y_{27} & u_{20} &= y_2 + y_4 + y_7 + y_{11} + y_{20} + y_{30}\\
u_5 &= y_2 + y_5 + y_6 + y_{15} + y_{28} & u_{21} &= y_0 + y_3 + y_5 + y_{12} + y_{21} + y_{31}\\
u_6 &= y_6 + y_{16} + y_{29} & u_{22} &= y_0 + y_1 + y_4 + y_{13} + y_{22}\\
u_7 &= y_7 + y_{17} + y_{30} & u_{23} &= y_1 + y_2 + y_5 + y_{14} + y_{23}\\
u_8 &= y_7 + y_8 + y_{18} + y_{31} & u_{24} &= y_2 + y_{15} + y_{24}\\
u_9 &= y_6 + y_9 + y_{19} & u_{25} &= y_7 + y_{16} + y_{25}\\
u_{10} &= y_7 + y_{10} + y_{20} & u_{26} &= y_0 + y_{17} + y_{26}\\
u_{11} &= y_0 + y_{11} + y_{21} & u_{27} &= y_1 + y_{18} + y_{27}\\
u_{12} &= y_1 + y_{12} + y_{22} & u_{28} &= y_2 + y_{19} + y_{28}\\
u_{13} &= y_2 + y_{13} + y_{23} & u_{29} &= y_3 + y_{20} + y_{29}\\
u_{14} &= y_6 + y_{14} + y_{24} & u_{30} &= y_0 + y_4 + y_7 + y_8 + y_{21} + y_{30}\\
u_{15} &= y_7 + y_{15} + y_{25} & u_{31} &= y_0 + y_1 + y_5 + y_9 + y_{22} + y_{31}
\end{aligned}$$

A.2 Masterkey Bits vs. Round Keys

Table 2. A list of master key bits used to generate the round keys in rounds 1 up to 10

round	master key bits used for the addition	bits input to the S-box
1	$\{0, \ldots, 31\}$	32 33 34 35 36 37 38 39
2	$\{39, \ldots, 55\} \cup \{0, \ldots, 14\}$	15 16 17 18 19 20 21 22
3	$\{22, \ldots, 53\}$	54 55 0 1 2 3 4 5
4	$\{5, \ldots, 36\}$	37 38 39 40 41 42 43 44
5	$\{44, \ldots, 55\} \cup \{0, \ldots, 19\}$	20 21 22 23 24 25 26 27
6	$\{27, \ldots, 55\} \cup \{0, 1, 2\}$	3 4 5 6 7 8 9 10
7	$\{10, \ldots, 41\}$	42 43 44 45 46 47 48 49
8	$\{49, \ldots, 55\} \cup \{0, \ldots, 24\}$	25 26 27 28 29 30 31 32
9	$\{32, \ldots, 55\} \cup \{0, \ldots, 7\}$	8 9 10 11 12 13 14 15
10	$\{15, \ldots, 46\}$	47 48 49 50 51 52 53 54

Message Authentication Codes from Unpredictable Block Ciphers

Yevgeniy Dodis[1] and John Steinberger[2]

[1] Department of Computer Science, New York University
dodis@cs.nyu.edu
[2] Department of Mathematics, University of British Columbia
jpsteinb@gmail.com

Abstract. We design an efficient mode of operation on block ciphers, SS-NMAC. Our mode has the following properties, when instantiated with a block cipher f to yield a variable-length, keyed hash function H:

(1) **MAC Preservation.** H is a secure message authentication code (MAC) *with birthday security*, as long as f is *unpredictable.*
(2) **PRF Preservation.** H is a secure pseudorandom function (PRF) *with birthday security*, as long as f is *pseudorandom.*
(3) **Security against Side-Channels.** As long as the block cipher f does not leak side-channel information about its internals to the attacker, properties (1) and (2) hold even if the remaining implementation of H is *completely leaky.* In particular, if the attacker can learn the transcript of all block cipher calls and other auxiliary information needed to implement our mode of operation.

Our mode is the first to satisfy the MAC preservation property (1) with birthday security, solving the main open problem of Dodis et al. [7] from Eurocrypt 2008. Combined with the PRF preservation (2), our mode provides a hedge against the case when the block cipher f is more secure as a MAC than as a PRF: if it is false, as we hope, we get a secure variable-length PRF; however, even if true, we still "salvage" a secure MAC, which might be enough for a given application.

We also remark that no prior mode of operation offered birthday security against side channel attacks, even if the block cipher was assumed pseudorandom.

Although very efficient, our mode is three times slower than many of the prior modes, such as CBC, which do not enjoy properties (1) and (3). Thus, our work motivates further research to understand the gap between unpredictability and pseudorandomness of the existing block ciphers, such as AES.

1 Introduction

Most primitives in symmetric-key cryptography are built from block ciphers, such as AES. Typically, one models the block cipher as a fixed-input-length (FIL) *pseudorandom* permutation (PRP), and then builds a more complex variable-input-length (VIL) primitive under this assumption. For many such VIL primitives, like pseudorandom functions (PRFs), this strong assumption on the block cipher is justifiable. One exception here is the design of message authentication codes (MACs): since the resulting primitive only needs to be *unpredictable*, it would be highly desirable to only assume that the block cipher is unpredictable as well, as opposed to pseudorandom. Indeed, it seems that assuming the block cipher is unpredictable is a *much weaker* assumption than assuming full pseudorandomness: to break the latter, all one needs to do is to predict one bit of "random-looking" information about the block cipher with probability just a little over 1/2, while the former requires one to fully compute the value of the block cipher on a new point with non-trivial probability. Thus, it is natural to ask the following central question of this work:

Question 1. Can one build an efficient variable-input-length MAC from a block cipher which is modeled as an *unpredictable* permutation (UP) on n-bits?

We will argue that no existing constructions are really satisfactory for achieving this natural goal. In order to discuss this precisely, we briefly recall some key quantities which determine the security of a construction. In this paper we consider only two types of adversaries: *distinguishers*, whose goal is to distinguish a function from an ideal primitive (as for PRFs and PRPs) and *forgers*, whose goal is to predict the value of the function on an un-queried message (as for MACs and UPs). As there often exist constant-query attacks using very long messages, the most important measure of an adversary's efficiency is the total length of messages that it queries. This upper bounds, among others, the number of queries made by the adversary. For functions that are built from a smaller primitive (such as, in all the cases we consider, a permutation), a more convenient efficiency measure is the number of queries one must make to the smaller primitive in order to evaluate the adversary's queries. In this section we let q stand for the latter number, as opposed to the number of queries actually made by the adversary to its oracle.

Let C be a function built from a block cipher f. The *security*[1] $\varepsilon = \varepsilon(q)$ of C is the maximum advantage of an adversary for which the number of calls to f necessary to compute the adversary's queries to C does not exceed q. Thus, lower ε implies better security. We write $\varepsilon_{\mathrm{mac}}$ and $\varepsilon_{\mathrm{prf}}$ to distinguish the security C as MAC and PRF, respectively. Likewise the block cipher has a security $\varepsilon_{\mathrm{up}}$ and $\varepsilon_{\mathrm{prp}}$ as a UP and as PRP, respectively.[2] The *rate* of a VIL-function C is the number of times the block cipher has to be called on each input block, so it measures the efficiency of C. We can now rephrase our goal as follows. Given a block cipher f with UP security $\varepsilon_{\mathrm{up}}$, construct a VIL-MAC C such that:

(a) C has small constant rate;
(b) the security $\varepsilon_{\mathrm{mac}}$ of C is as small as possible as a function of $\varepsilon_{\mathrm{up}}$.

A good way to quantify the "goodness" of $\varepsilon_{\mathrm{mac}}$ is to assess the maximum q for which the achieved bound is meaningful, assuming that the block cipher has ideal

[1] Other parameters, such as the running time allowed to the adversary, may be relevant for the security, but these are not important here.

[2] In fact $\varepsilon_{\mathrm{up}} = \varepsilon_{\mathrm{mac}}$ for a block cipher, since these refer to the same notion, but we write $\varepsilon_{\mathrm{up}}$ to emphasize the difference with C.

security $\varepsilon_{\mathrm{up}} \sim 1/2^n$ as an unpredictable permutation. For example, if $\varepsilon_{\mathrm{mac}} = O(\varepsilon_{\mathrm{up}} \cdot q^2)$, then the bound is meaningful for q up to $\sim 2^{n/2}$, which matches the classical birthday bound typically achieved when one models the block cipher as a PRP. As argued by Preneel and van Oorschot [15], a simple extension attack shows that the birthday security is the best security one may hope to achieve by natural "iterative" constructions. On a positive, several elegant (iterative) constructions matching this bound are known, when modeling the block cipher as a PRP. On a negative, no existing constructions, iterative or otherwise, come even close to the birthday security when assuming UPs as opposed to PRPs. Thus, our "golden standard" to answer Question 1 will be to solve.

Question 2. Build an (iterative) VIL-MAC from UPs, having constant efficiency rate and (nearly) birthday security.

Jumping ahead, this will be our main result, therefore resolving the main open question of Dodis et al. [7] from Eurocrypt 2008. But first, let us survey what is known, to better understand the difficulties we will have to face, and also motivate our approach.

1.1 Inapplicability of Existing Solutions

There is a huge number of proposals for building a VIL-MAC out of a block cipher. Unfortunately, it turns out that most of them are insecure when instantiated with unpredictable block ciphers, — often despite having simple proofs of security when one models the block cipher as a PRP, — and the few that are secure, achieve extremely poor rate and/or security. In the full version [9], we give a detailed listing of many "failed" approaches to build an efficient VIL-MAC from an unpredictable block cipher. Here, we will only give a brief summary, concentrating on the approaches most relevant to our eventual solution.

In brief, the existing approaches in question include the following: (1) generic route from unpredictability to pseudorandomness [10, 13]; (2) CBC-MAC [4, 14]; (3) HMAC/NMAC [6, 3]; (4) various ad-hoc methods (e.g., iterating the truncated version of the block cipher); (5) hash-then-MAC using (almost) universal hashing [5, 3]; (6) hash-then-MAC using collision-resistant hashing; (7) Feistel Network [11,8]; and (8) the current best method called "enhanced CBC" mode [7]. Of these, approaches (2), (3), (4) and (5) are completely insecure when instantiated with generic UPs (as opposed to PRPs!). This is simple to see for (3), (4) and (5), and was shown by An and Bellare [2] for the CBC-MAC (approach (2)). The generic approach (1) is secure, but very inefficient, which is not surprising.

Approach (6), using a collision-resistant hash function (CRHF) H to hash the VIL message before applying a FIL-MAC, also works in principle, but is not satisfactory. In theory, the assumption that CRHFs exist is much stronger than the existence of UPs (or even PRPs); for example, there is a black-box separation [18] between these assumptions. Even in practice, where many hash functions are built from block ciphers, the resulting hash functions appear to require a larger security parameter than the "stand-alone" block ciphers they are built from. For example, while the industry standard AES has input length 128, no existing hash function with 128-bit output is considered secure (e.g., MD5 and related functions are broken [19]); in fact, NIST does not recommend using any hash function with output size below 256, including 160-bit SHA-1.

Weak Collision-Resistance. Thus, we would like to base security of the "hash-then-mac" approach on weaker hash functions than CRHFs. As was demonstrated by [6], the precisely correct notion for this task is that of *Weak Collision Resistance* (WCR). Such hash functions H are keyed, and their key is part of the secret key for the resulting VIL-MAC. In terms of security, it should be infeasible for the attacker to come up with distinct inputs x and y such that $H(x) = H(y)$, *even when given oracle access to* H. An and Bellare [2] then showed that WCR hash functions have similar properties to CRHFs: in particular, the (strengthened) Merkle-Damgard transform gives a VIL-WCR hash function from a FIL-WCR hash function, which can then be used in "hash-then-mac" approach. Moreover, both security reductions are tight for our purposes.

Thus, to efficiently answer Question 1, *it is sufficient to build a fixed-input-length sufficiently compressing (say, two-to-one) WCR hash family.* Indeed, this is the route of all existing solutions (e.g., approaches (7) and (8)), as well as our solution. However, to also answer Question 2 would additionally require a WCR hash *with birthday security*, which was not known prior to this work.

Building WCR from UP. This question appears to be non-trivial. In particular, only two secure solutions were known prior to this work (approaches (7) and (8) above). First, Dodis and Puniya [8] showed how to construct a two-to-one FIL-WCR from $\omega(\log \kappa)$ independent UPs, where κ is the security parameter. The construction applied $\omega(\log \kappa)$ rounds of the Feistel Network $\pi(x\|y) = (y\|f(y) \oplus x)$ to the $2n$-bit input, each with a different UP f, and then truncated the last output in half. Moreover, they showed that $O(\log \kappa)$ rounds are generally insufficient for this task (extending the three-round counter-example of [2], and in sharp contrast to the setting of PRPs, where three rounds are already enough [11]). This means that the resulting super-constant rate $\omega(\log \kappa)$ of this particular construction cannot be improved, making it somewhat inefficient for practice. More significantly, the security of this construction proven by [8] was only $O(\varepsilon_{\mathrm{up}} \cdot q^6)$, meaning that it can only be secure for at most $2^{n/6}$ messages, which is unacceptable for $n = 128$.

The best current WCR construction from UPs comes from the work of Dodis et al. [7], who made the surprisingly simple observation that the function $h(x\|y) = f_1(x) \oplus f_2(y)$ is a two-to-one, rate-2 WCR hash function, assuming f_1 and f_2 are two independent UPs. This immediately gives a rate-2 VIL-MAC from UPs, which is very efficient, and is the first (and only) constant-rate solution to Question 1 known prior to this work. Unfortunately, the security of this WCR function (and the resulting VIL-MAC) is $O(\varepsilon_{\mathrm{up}} \cdot q^4)$. Moreover, it is easy to see that this bound is actually tight. Thus, the construction can only be secure for at most $2^{n/4}$ messages, again making it fall short of our goal of obtaining security up to $2^{n/2}$. In fact, this question of achieving "birthday security" $2^{n/2}$ (which is our Question 2) was the main open question posed in [7].

1.2 Our Results

In this work we resolve this question in the affirmative. Concretely, we construct a VIL-MAC, called SS-NMAC, from four independent UPs $f_1, \ldots, f_4$, which achieves rate 3 and security $\varepsilon_{\mathrm{mac}} \approx O(\varepsilon_{\mathrm{up}} q^2 \log^2(q))$, meaning it can be secure for almost $2^{n/2}$ messages. This is the first constant-rate MAC with birthday security built from an unpredictable block cipher. The construction of SS-NMAC is depicted in Figure 2, where the message is $x = x_1 \ldots x_\ell$.

Our construction uses the WCR approach mentioned earlier: namely, it uses the (strengthened) Merkle-Damgard iteration of the $2n$-bit to n-bit compression function $F(x\|y) = f_1(x) \oplus f_3(f_1(x) \oplus f_2(y))$, which is shown in Figure 1. This function was originally suggested by Shrimpton and Stam [17], who argued that F is collision-resistant with birthday security, assuming that f_1, f_2, f_3 are *public random functions* (i.e., random oracles). In contrast, our main technical result (Theorem 1) shows that this function is (weakly) collision-resistant (with birthday security), even if f_1, f_2, f_3 are only (keyed) *unpredictable* functions.

We note that since any FIL-MAC is FIL-WCR (Lemma 4.4 [2]) it would suffice to prove the Shrimpton-Stam compression function is a good MAC in order to show it is WCR. However, as explained in the full version [9], the Shrimpton-Stam compression function is not a good enough MAC for our purposes, showing the necessity of directly proving WCR security.

Comparison with [17]. On a technical level, both results appear similar. In both cases, assuming the adversary A has oracle access to f_1, f_2, f_3, one has to argue that A has low chance of finding a collision to F. However, the key difference is that the f_i's are assumed truly random in [17], whereas we can only assume *unpredictable* f_i's. In particular, while [17] could directly bound the probability of A finding the collision in F using an *information-theoretic argument*, we have to *build an efficient reduction* from the presumed collision-finding attacker A to a UP-forger B forging one of the MACs. It is well known that such information-theoretic arguments often do not have direct analogs in the computational setting. To illustrate this more concretely, let us only discuss the most interesting such difficulty we had to resolve.

The key argument of [17] was a technical calculation, using factorials, binomials and various probability manipulations, that A is unlikely to find an "n-way multi-collision" in the auxiliary function $h(x\|y) = f_1(x) \oplus f_2(y)$, when f_1 and f_2 are truly random functions. To adapt this (critical) part of the argument to our computational setting, we would have to take an efficient attacker A' capable of finding such a multi-collision in h with probability ε, and turn it into a forger B' for either f_1 or f_2, succeeding with probability $\Omega(\varepsilon/q^2)$. As far as we could see, the probability calculations in [17] give no guidance of how to do such a reduction. And, indeed, finding such a reduction required a completely new approach, relating to a natural "balls-and-bins" game that we analyzed (see Lemma 1), and resulting in a very non-obvious construction of B. We discuss this construction in detail in Section 4.2, only mentioning that it gave us a better understanding of the security of the Shrimpton-Stam compression function, and even implicitly improved the probability calculations of [17] for the special case of truly random functions (corresponding to $\varepsilon = 2^{-n}$ in our reduction). In particular, for the case of random functions we get a convenient closed form of $O(q^2 \log^2(q)/2^n)$ for the collision resistance of the Shrimpton-Stam compression function, where, as per our convention for this section, q is the total number of block cipher queries allowed (cf. Theorem 1).

Strong PRF preservation. We also notice that our new mode has the following desirable multi-property preservation guarantee advocated by [7]: if the block cipher is unpredictable, we get a MAC with message security roughly $2^{n/2}$, while if it happens to be pseudorandom, we get a PRF with message security roughly $2^{n/2}$. In other words, we expect and hope that practical block ciphers (such as AES) are in fact PRPs with good security. If our hope is correct, we would get a full-fledged pseudorandom function with good security; however, even if the block cipher turns out to be a much better MAC than it is a pseudorandom function, we still get a MAC with excellent security, which could be reassuring for many applications. Details are sketched in Section 5.

More interestingly, even in the setting of PRPs, *our SS-NMAC construction yields a more "leakage-resilient" VIL-PRF H than the prior constructions.* In particular, in Section 6 we show that the resulting PRF is secure even in the so called *oracle cipher model*, first considered by Dodis et al. [8]. Recall, in the standard model for PRFs, the attacker only learns the output $H(x)$ of the PRF on input x, but does not learn any of the intermediate values, such as the inputs/outputs to the block cipher or any of the chaining variables. Indeed, this secrecy of the intermediate values is completely essential to the security of most standard constructions, such as CBC-MAC or the standard Luby-Rackoff transformation [11]. In other words, these constructions are actually *broken* in the oracle cipher model, irrespective of the strength of the block cipher used (e.g., even with AES). In contrast, our SS-NMAC construction is a secure VIL-PRF — *with (essentially) the same birthday security* — even when the attacker *learns all the intermediate values needed to obtain $H(x)$*, except for what is done inside the actual block-cipher computations. More precisely, even if the attacker learns all the computation history of our SS-NMAC construction on a bunch of points (not including the internals of the block ciphers), the value of the function at any set of *non-queried* points looks random to the attacker. Thus, as long as the block cipher is implemented in a "leakage-resilient" way, the remaining implementation of SS-NMAC can be completely insecure with respect to side-channel attacks! We believe that the security in the oracle model is quite important, since we envision secure *hardware-based* implementation of block-ciphers, later composed with much less secure *software-based* solutions, to yield various more advanced VIL primitives. We also remark that none of the two previous PRF constructions in the oracle cipher model [8, 7] achieved anything close to birthday security.

Summary. To summarize, in addition to yielding a more secure VIL-MAC than prior constructions in the case when $\varepsilon_{\rm up} \ll \varepsilon_{\rm prp}$, our construction gives a more "leakage-resilient" (and equally secure!) VIL-PRF even when assuming $\varepsilon_{\rm prp}$ is nearly as good as $\varepsilon_{\rm up}$. Moreover, we only pay a small constant factor price in efficiency for these (significant) security enhancements. In Section 7, we briefly discuss whether this slowdown is justifiable in practice, which ultimately calls for more research to understand the gap between unpredictability and pseudorandomness of the existing block ciphers, such as AES.

2 Security Definitions

We briefly recall the standard security notions for MACs and PRFs. In each case we are interested in resistance to chosen message attacks. For a MAC, an adversary succeeds if it can forge the MAC on an un-queried value. For a PRF, the adversary succeeds if it can distinguish the PRF from a truly random oracle. To measure an adversary's efficiency we count not only the number of oracle queries made but also the time and the total length of queried messages (as the oracles accept variable length inputs). In this section we use the variable $\tilde{q}$ to denote the number of queries made by the adversary to its oracle in order to emphasize the distinction from the variable q used in Section 1, which was defined as the (distinct) number of block cipher calls necessary to evaluate those queries (the adversary does not have direct access to the block cipher). In later sections we maintain the spirit of this convention, using $\tilde{q}$ for queries made to VIL-functions and q for queries made to FIL-functions (usually block ciphers).

A *function family* is a map $f : \{0,1\}^\kappa \times Dom(f) \to \{0,1\}^n$ where $Dom(f) \subseteq \{0,1\}^*$. The strings in $\{0,1\}^\kappa$ are the *keys* of f and we write $f_k(x)$ for $f(k,x)$ for $k \in \{0,1\}^\kappa$ and $x \in Dom(f)$. The function f_k is called a *member* of f.

For MACs we consider the following game, where A is an adversary with oracle access to f_k:

```
Game Forge($A, f$):
  $k \leftarrow \{0,1\}^\kappa$; $(x, y) \leftarrow A^{f_k}$
  If $x \in Dom(f)$, $f_k(x) = y$ and $x$ was not a query of $A$ then $A$
    wins, otherwise $A$ looses.
```

Following An and Bellare [2] we define the insecurity of f as a MAC to be

$$\mathbf{InSec}_f^{\mathrm{mac}}(t, \tilde{q}, \mu) := \max_A \Pr[A \text{ wins Forge}(A, f)]$$

where the maximum is taken over all adversaries A making at most $\tilde{q}$ queries whose total combined length is at most μ bits and of "running time" at most t. The "running time" is defined to be the total running time of the experiment, including the time necessary to compute the answers to A's queries. (The advantage of this definition is that a simulator running A and computing the answer to A's queries from scratch has essentially the same running time t.)

For PRF security the game is modified by either giving A access to a random f_k or to a random oracle $g : Dom(f) \to \{0,1\}^n$ with probability $\frac{1}{2}$ and A wins if it correctly identifies whether its oracle is f_k or g. Call this game 'Identifies(A, f)'. Then

$$\mathbf{InSec}_f^{\mathrm{prf}}(t, \tilde{q}, \mu) := \max_A \Pr[A \text{ wins Identifies}(A, f)] - \frac{1}{2}$$

where again the maximum is taken over all adversaries A making at most $\tilde{q}$ queries of total length μ and of running time t, with the same convention concerning running time.

The proof finally uses the notion of "weak collision resistance" (WCR), which measures the collision resistance of a function only available as an oracle to the adversary. In the weak collision resistance game for the function family f, A is given oracle access to a random f_k and wins if it succeeds in querying f_k on two distinct points x, y such that $f_k(x) = f_k(y)$. Then $\mathbf{InSec}_f^{\mathrm{wcr}}(t, \tilde{q}, \mu)$ is defined similarly with respect to this game as $\mathbf{InSec}_f^{\mathrm{mac}}(t, \tilde{q}, \mu)$ is defined with respect to the game $\mathrm{Forge}(A, f)$.

3 The SS-NMAC Construction

The basic SS-NMAC scheme is shown in Figure 2. The scheme uses the Merkle-Damgard iteration of the $2n$-bit to n-bit compression function of Shrimpton and Stam [17] shown in Figure 1. We start by describing this compression function.

The Shrimpton-Stam compression function. The Shrimpton-Stam compression function is a $2n$-bit to n-bit compression function that uses calls to three different n-bit to n-bit primitives f_1, f_2, f_3. We write the compression function as $F[f_1, f_2, f_3]$ to emphasize its dependence on f_1, f_2, f_3. It is defined by

$$F[f_1, f_2, f_3](x\|y) = f_1(x) \oplus f_3(f_1(x) \oplus f_2(y))$$

for any pair of n-bit strings x, y.

Shrimpton and Stam [17] proved that $F[f_1, f_2, f_3]$ has optimal (i.e. birthday) collision resistance if f_1, f_2, f_3 are random functions. They also conjectured that the construction remains collision resistant if $f_i(x)$ is replaced with $\pi_i(x) \oplus x$ where π_1, π_2, π_3 are random permutations, which would enable the construction to be implemented with fixed key block ciphers. This conjecture was verified by Rogaway and Steinberger [16].

For our purposes, the key property of $F[f_1, f_2, f_3]$ is that an adversary with oracle access to the f_i's can only learn $F[f_1, f_2, f_3](x\|y)$ on roughly as many inputs $x\|y$ as it makes queries. This should be contrasted for example to the compression function $h[f_1](x\|y) = f_1(x \oplus y)$ of the CBC MAC or the "xor

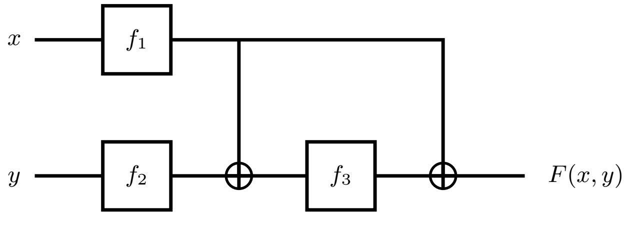


Fig. 1. The Shrimpton-Stam compression function. All wires carry n-bit values.

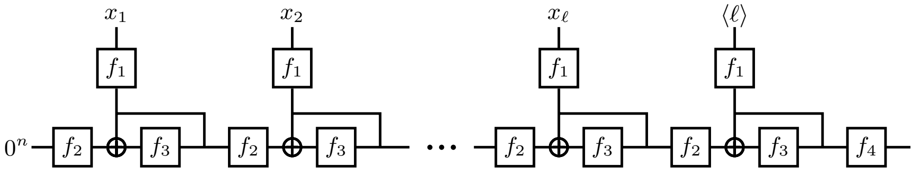


Fig. 2. The SS-NMAC mode of operation

compression function" $g[f_1, f_2](x\|y) = f_1(x) \oplus f_2(y)$ of the enciphered CBC construction of Dodis, Pietrzak and Puniya [7], where f_1, f_2 are again n-bit to n-bit functions. An adversary querying $h[f_1]$ can learn to evaluate $h[f_1]$ on 2^n inputs $x\|y$ in a single query; an adversary querying $g[f_1, f_2]$ can learn to evaluate $g[f_1, f_2]$ on q^2 inputs $x\|y$ in q queries. Another compression function that could be used equally well in place of $F[f_1, f_2, f_3]$ is the LP231 compression function of Rogaway and Steinberger [16], which also uses three calls to n-bit to n-bit primitives. However we use $F[f_1, f_2, f_3]$ because it is simpler and sufficient for our purposes.

Iteration and Padding. First we define $\mathrm{PadAp}(x)$ to be $x10^k\langle \ell \rangle$ where k is the least integer such that $x10^k$ has length a multiple of n, where ℓ is the number of n-bit blocks in $x10^k$, and where $\langle \ell \rangle$ is ℓ written as an n-bit binary integer (messages with maximum length 2^n are sufficient for most applications). Appending $\langle \ell \rangle$ amounts to using Merkle-Damgard strengthening, which we do in order to keep our space of messages suffix-free. Any other suffix-free encoding of messages would do as well.

To iterate $F[f_1, f_2, f_3]$ we define the "SS-cascade" $G[f_1, f_2, f_3]$ of an $n\ell$-bit string $x = x_1\|\cdots\|x_\ell$ where each x_i is an n-bit string by $G[f_1, f_2, f_3](x) = y_\ell$ where $y_0 = 0^n$ and $y_k = F[f_1, f_2, f_3](x_k\|y_{k-1})$ for $1 \leq k \leq \ell$. Finally, given an additional n-bit to n-bit function f_4 we define the SS-NMAC $H[f_1, f_2, f_3, f_4]$ by

$$H[f_1, f_2, f_3, f_4](x) = f_4(G[f_1, f_2, f_3](x)).$$

for all $x \in \mathrm{Dom}(H) := \{\mathrm{PadAp}(y) : y \in \{0,1\}^*\}$. See Figure 2, where $x = x_1\|\cdots\|x_\ell\|\langle \ell \rangle$. Note that to query $H[f_1, f_2, f_3, f_4]$ on its domain an adversary must pad the input itself before giving it to the oracle. Thus queries must be at least n bits long and the number of queries made by an adversary is upper bounded by μ/n where μ is the total length of messages queried by the adversary.

For the remainder of the paper we let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be an arbitrary, fixed function family. We consider H as a function family of signature $\{0,1\}^{4\kappa} \times \mathrm{Dom}(H) \to \{0,1\}^n$, where $H_{k_1k_2k_3k_4}(x) := H[f_{k_1}, f_{k_2}, f_{k_3}, f_{k_4}](x)$. Likewise we consider F as a function family of signature $\{0,1\}^{3\kappa} \times \{0,1\}^{2n} \to \{0,1\}^n$ defined by $F_{k_1k_2k_3}(x\|y) = F[f_{k_1}, f_{k_2}, f_{k_3}](x\|y)$.

4 Security of SS-NMAC as a MAC

4.1 Overview

In this section we outline the proof that SS-NMAC is a secure MAC when $f_1, \ldots, f_4$ are secure MACs. The proof shows that H is a secure MAC family if f is a secure MAC family. In fact,

$$\mathbf{InSec}_H^{\mathrm{mac}}(t, \tilde{q}, \mu) \leq \left(1 + 30q^2 \log^2(q)\right) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(t + O(q^2 n), q, qn) \qquad (1)$$

where $q = \mu/n$ ($\tilde{q}$ is inconsequent, though one automatically has $\tilde{q} \leq q$). The $O(q^2 n)$ difference in running time is due to the overhead of a simulator.

Like Dodis, Pietrzak and Puniya [7], our security proof follows the approach developed by An and Bellare [2], who reduce the VIL-MAC security to FIL-WCR security. In order to summarize their method in a convenient way we refer to the members of a function family as being MAC-secure or WCR-secure (see section 2 for the definition of WCR security) though security is really a property of the function family. An and Bellare reduce the MAC security of a VIL function to the WCR security of a FIL function in two steps:

Step 1: The composition of a secure FIL-MAC f_k and a secure WCR function $G_{k'}$ is a secure VIL-MAC $f_k(G_{k'}(\cdot))$ (Lemma 4.2 [2]). Applying this to the case where $f_k = f_{k_4}$ and $G_{k'} = G[f_{k_1}, f_{k_2}, f_{k_3}]$ it therefore suffices to show that $G[f_{k_1}, f_{k_2}, f_{k_3}]$ is WCR-secure if f is a secure MAC family in order to show that $H[f_{k_1}, f_{k_2}, f_{k_3}, f_{k_4}] = f_{k_4}(G[f_{k_1}, f_{k_2}, f_{k_3}])$ is a secure MAC family.

Step 2: On a suffix-free domain of inputs the Merkle-Damgard iteration of a FIL-WCR compression function gives a VIL-WCR function (Lemma 4.3 [2]). Thus, by step 1, it suffices to show that the Shrimpton-Stam compression function $F[f_{k_1}, f_{k_2}, f_{k_3}]$ is FIL-WCR when f is a secure MAC family.

Steps 1 and 2 give a qualitative description of An and Bellare's approach. Quantitatively, their Lemmas 4.2 and 4.3 imply that

$$\mathbf{InSec}_H^{\mathrm{mac}}(t, \tilde{q}, \mu) \leq \mathbf{InSec}_f^{\mathrm{mac}}(t, q, qn) + \mathbf{InSec}_F^{\mathrm{wcr}}(t, q, 2qn) \qquad (2)$$

where $q = \mu/n$. Since $\mathbf{InSec}_f^{\mathrm{mac}}(t, q, qn) \leq \mathbf{InSec}_f^{\mathrm{mac}}(t + O(q^2 n), q, qn)$ it therefore suffices to prove

$$\mathbf{InSec}_F^{\mathrm{wcr}}(t, q, 2qn) \leq 30q^2 \log^2(q) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(t + O(q^2 n), q, qn) \qquad (3)$$

in order to prove (1). Inequality (3) is really the paper's main result, and we state it as a theorem:

Theorem 1. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ and let $F : \{0,1\}^{3\kappa} \times \{0,1\}^{2n} \to \{0,1\}^n$ given by $F_{k_1 k_2 k_3}(x \| y) = F[f_{k_1}, f_{k_2}, f_{k_3}](x \| y) = f_{k_1}(x) \oplus f_{k_3}(f_{k_1}(x) \oplus f_{k_2}(y))$. Then*

$$\mathbf{InSec}_F^{\mathrm{wcr}}(t, q, 2qn) \leq 30q^2 \log^2(q) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(t + O(q^2 n), q, qn).$$

The full proof of Theorem 1 is proven in the full version [9], but we give an outline in the next section.

Together with Lemmas 4.2 and 4.3 of [2], Theorem 1 implies inequality (3), which we restate as our theorem characterizing the MAC security of SS-NMAC:

Theorem 2. *Let $f : \{0,1\}^{\kappa} \times \{0,1\}^n \to \{0,1\}^n$ and let $H : \{0,1\}^{4\kappa} \times \mathrm{Dom}(H) \to \{0,1\}^n$ be the SS-NMAC function family. Then, letting $q = \mu/n$,*

$$\mathbf{InSec}_H^{\mathrm{mac}}(t, \tilde{q}, \mu) \le \big(1 + 30q^2 \log^2(q)\big) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(t + O(q^2 n), q, qn).$$

4.2 Proof Outline

In this section we give a proof of Theorem 1 under several simplifying assumptions, which make our presentation considerably easier, while maintaining the key ideas of the full proof. Recall, we need to upper bound the WCR-insecurity of the Shrimpton-Stam compression function F in terms of the MAC-insecurity of f. Equivalently, we must lower bound the MAC-insecurity of f in terms of the WCR-insecurity of F. To do the latter, we show how an ε-collision-finding adversary A for F can be turned into a δ-MAC-forging adversary B for f, where B uses the same number of queries but has chance of success $\delta = \Omega(\varepsilon/q^2 \log^2(q))$.

First, instead of giving A oracle access to $F[f_1, f_2, f_3]$, we directly give it oracle access to f_1, f_2, f_3, with q queries allowed to each f_i. Clearly, such an adversary can simulate q queries to F, so we only made A more powerful. (Note, this strengthened attacker will be useful when we extend our argument to the "oracle cipher" model in Section 6.) Let us generally denote the inputs to f_1, f_2, f_3 by x, y, z, respectively, and also denote by $x_1 \dots x_q$, $y_1 \dots y_q$ and $z_1 \dots z_q$ the ordered inputs to f_1, f_2, f_3 supplied by A. As expected, the forger B will simulate this adversary A when trying to forge one of the f_i's, by using its own oracle to simulate the corresponding f_i, and simulating the other f_j's by picking their secret keys by itself and answering honestly.

Simplifying Assumptions. Before proceeding further, we state our simplifying assumptions on the behavior of A, which will make our construction of B much simpler, while retaining the key ingredients of the general case.

- **(No Collision in f_i's)** For any distinct inputs x_r and x_s that $\mathcal{A}$ supplied to f_1, $f_1(x_r) \neq f_1(x_s)$. Similar conditions also hold for f_2 and f_3.
- **(Query Order)** All the calls to f_1 and f_2 are made by A before any call to f_3 is made.

Let us briefly comment on these assumptions. The first assumption regarding the collisions in the f_i's is very minor, and is done for convenience only. Indeed, in the actual applications, the f_i's are *permutations*, so the assumption is trivially true. And even if the f_i's are arbitrary length-preserving MACs, the failure to satisfy our assumption with probability $\Omega(\varepsilon)$ trivially leads to a simple attacker B, forging the corresponding f_i with probability $\Omega(\varepsilon/q^2)$, by simply guessing the indices $r, s \in \{1 \dots q\}$ of the colliding queries. So the only "real" assumption we make is the **Query Order** Assumption. This assumption is provably impossible for the "initial" attacker who has oracle access to $F[f_1, f_2, f_3]$, as opposed to f_1, f_2, f_3 (since f_3 will be called on the first call there), and is even more unreasonable for the generalized attacker that can query the f_i's in any order it wants. However the assumption is used in a rather weak way in the proof sketch, as we will see, so that eliminating it only requires additional casework, and no significant new ideas.

Notation and Terminology. A *ball* is a pair (x, y) where $x, y \in \{0,1\}^n$. A *bin* is a value $z \in \{0,1\}^n$. It is instructive to associate balls (x, y) with the inputs to F, and the bins z with the inputs to f_3. After A makes q queries $x_1 \ldots x_q$ to f_1 and $y_1 \ldots y_q$ to f_2, we get $Q = q^2$ potential balls (x_r, y_s) "thrown" by A. In particular, we will say that such (x_r, y_s) is *placed* into the bin $z = f_1(x_r) \oplus f_2(y_s)$, and let $\mathsf{Bin}(z) = \{(x_r, y_s) : f_1(x_r) \oplus f_2(y_s) = z\}$ denote the set of balls placed into bin z. Notice, each query x_r to f_1 allows the attacker to simultaneously place $j \leq q$ balls $(x_r, y_1), \ldots, (x_r, y_j)$, where j is the number of queries to f_2 made so far. However, under our **No Collision** assumption of f_2, all these j balls go to *distinct* bins $f_1(x_r) \oplus f_2(y_s)$, where $1 \leq s \leq j$. Similar discussion holds for the calls to f_2. Also, under our **Query Order** Assumption, the attacker A places all Q balls into the appropriate bins in the first stage, before making any of its queries $z_1 \ldots z_q$ to f_3 in the second stage. And after each such query z_t to f_3, A learns the value of $F(x\|y) = f_1(x) \oplus f_3(z_t)$ *precisely for all* $(x, y) \in \mathsf{Bin}(z_t)$.

Back to Reduction. By our assumption, A will find a collision $(x, y) \neq (x', y')$ to F with probability ε. Without loss of generality, we assume that A makes the queries necessary to verify this collision. Thus, $x, x' \in \{x_1, \ldots, x_q\}$, $y, y' \in \{y_1, \ldots, y_q\}$, and $z, z' \in \{z_1, \ldots, z_q\}$, where $z = f_1(x) \oplus f_2(y)$ and $z' = f_1(x') \oplus f_2(y')$. Notice, under our **No Collision** assumption on f_1 and f_2, we claim that $z \neq z'$. Otherwise, $f_3(z) = f_3(z')$ and $f_1(x) \oplus f_3(z) = F(x\|y) = F(x'\|y') = f_1(x') \oplus f_3(z')$ imply that $f_1(x) = f_1(x')$, meaning that $x = x'$. Then $f_2(y) = f_1(x) \oplus z = f_1(x') \oplus z' = f_2(y')$, so $y = y'$, meaning that $(x, y) = (x', y')$. Hence, the "colliding" bins z and z' queried by A must be *distinct.*

We now define a key parameter which will determine the behavior of our forger B: *the maximum bin size* $m = \max_z |\mathsf{Bin}(z)|$ after the calls to f_1 and f_2 (the "filling" stage). We consider two complementary cases: (1) A finds a collision and $m \leq \log(q)$, meaning that every bin z contains at most $\log(q)$ balls after the calls to f_1 and f_2 are finished; and (2) $m > \log(q)$, meaning that A managed to produce more than $\log(q)$ pairs (x_r, y_s) resulting in the same value $z = f_1(x_r) \oplus f_2(y_s)$.

(Interestingly, this parameter m corresponds to the largest "multi-collision" generated by A in the "filling" stage. As argued by Shrimpton and Stam [17] for the case of *truly random* functions f_i and $q \approx 2^{n/2}$, the value m must be smaller than $n^{1+o(1)} \approx \log q$ with high probability, more or less corresponding to saying that the attacker A must almost always be in case (1).)

By assumption that A succeeds to find a collision with probability $\geq \varepsilon$, at least one of these complementary cases happens with probability $\geq \varepsilon/2$.

Case (1): *A finds a collision and* $m \leq \log(q)$. This is the "easy" case. Intuitively, by querying at most q bins z in the second stage, A learned the value of F in at most $qm \le q\log(q)$ points (x,y). As we will see, it will allow B to guess the colliding points $(x,y),(x',y')$ with probability $1/(q\log(q))^2$, and then forge the value $f_3(z') = f_3(z) \oplus f_1(x) \oplus f_1(x')$. More formally, B starts by choosing two random indices $j < i$ between 1 and q. Let z_i, z_j be the i-th and j-th queries made to f_3. When the query $f_3(z_i)$ is made, B chooses random elements $(x_i,y_i) \in \mathsf{Bin}(z_i)$ and $(x_j,y_j) \in \mathsf{Bin}(z_j)$ (assuming these sets are nonempty, otherwise B gives up), and predicts that $f_3(z_i) = f_1(x_i) \oplus f_1(x_j) \oplus f_3(z_j)$. Notice, this corresponds to guessing that $F(x_i,y_i) = F(x_j,y_j)$, which implies that A is about to find a collision. This strategy cannot be successful unless A finds a collision (which we are assuming happens in this case), and unless the colliding bins z_i and z_j are distinct, which we also argued earlier as following from our **No Collision** assumption. But when A does find a collision, B's chance of guessing the indices i, j correctly is $1/\binom{q}{2} \ge 1/q^2$. Moreover if $\max_z |\mathsf{Bin}(z)| \le \log(q)$, B's chance of guessing the right elements (x_i,y_i) and (x_j,y_j) in $\mathsf{Bin}(z_i)$ and $\mathsf{Bin}(z_j)$ is at least $1/\log(q)$ each. Thus B's chance of success with this strategy is at least $1/q^2\log^2(q)$ when $\max_z |\mathsf{Bin}(z)| \le \log(q)$ and A finds a collision.

Case (2): *A produces $m > \log(q)$.* This is the "hard" case, where our balls-and-bins terminology comes in handy. Intuitively, if A throws q^2 balls with a guarantee that some bin will contain a lot of balls at the end, B should have a non-trivial chance (analyzed below) to guess the bin z corresponding to some ball (x,y) *before this ball is thrown.* To effect such a guess, when A "throws" the ball (x,y) by querying, say, $f_1(x)$ after previously querying $f_2(y)$, B can predict that $f_1(x) = z \oplus f_2(y)$, or conversely with f_1 and f_2 reversed if A queries $f_2(y)$ after querying $f_1(x)$. In other words, predicting the output of f_1 or f_2 on a value queried by A is *equivalent* to predicting the bin where a particular ball (x,y) will land at the point when the latest of the two queries $f_1(x)$, $f_2(y)$ is made. Thus, we may view B's task as consisting of observing a set of $Q = q^2$ balls being placed by groups in 2^n bins, and interrupting the game at some point to *predict the bin where a particular ball that is about to be placed.* We model this by a "balls-and-bins" game played by A and B, where A is incrementally throwing Q balls into bins, trying to fill some bin with more than $\log(q)$ balls, and yet without having B be able to guess the position of a ball before it is placed. Based on our discussion, the precise "rules" of this game are as follows:

Balls-and-Bins Game:

– The game proceeds in $2q$ rounds, after which A is required to throw exactly $Q = q^2$ balls.
– Before each round, A announces to B at most q balls $b_1,\dots,b_t$ that it will be throwing into (necessarily) *distinct* bins in this round. [Intuitively, a round corresponds to a query to $f_1(x_r)$ (or $f_2(y_s)$), and the balls are the corresponding values (x_r,y_j) (or (x_i,y_s)) for prior x_i's or y_j's.]
– In turn, B can secretly "pass" or make a "guess" (ℓ,z) that the ball b_ℓ will be thrown into bin z (where $1 \le \ell \le t$). [Intuitively, a successful guess will allow B to forge either f_1 or f_2, as outlined earlier.]

- A announces to B the bins where $b_1 \dots b_t$ are thrown. [Intuitively, B learns the value of f_1 or f_2 at the queried point, allowing it to learn the bin identities.]
- If B made a guess during this round, B wins the game if the guess is correct, and loses otherwise. If B did not make a guess, proceed to the next round.
- B must make a guess at some round, while A must fill at least one bin with more than $(\log q)$ balls.

Lemma 1. *Irrespective of A's strategy, there exists an efficient strategy for B to win the above game with probability at least $1/4q^2$ whenever some bin contains more than $\log(q)$ balls at the end of the game.*

Proof. B's strategy is relatively simple:

1. Choose a random index i between 1 and q^2, and a second random integer k between 1 and $\log(q)$.
2. Pass in all the rounds before the i-th overall ball is about to be thrown.
3. When the i-th ball is about to be thrown, make a secret guess that this ball will be thrown in a random bin z chosen among those bins *already containing at least k balls prior to this round* (or guess any bin if no such bin exists).

We argue that with this strategy, B's chance of success is at least $1/4q^2$, provided that some bin contains more than $\log(q)$ balls by the end of the game. Let c_j be the total number of balls that are thrown into bins that already have at least j balls in them right before the round when this ball is thrown. Thus $c_0 = q^2$ and $c_{\log(q)} \geq 1$ by assumption that a "heavy" bin exists at the end of the game. Also note that c_j is an upper bound for the number of bins that have $j+1$ balls in them at the end of the game, since for a bin to receive $j+1$ balls, some ball has to be thrown into it when the bin already has j balls.

For a fixed value of k, B's chance of correctly guessing the bin is at least $\frac{c_k}{q^2} \cdot \frac{1}{c_{k-1}} = \frac{1}{q^2} \cdot \frac{c_k}{c_{k-1}}$. This is because B has chance at least $\frac{c_k}{q^2}$ of choosing a ball thrown into a bin with at least k balls, and then has at least chance $\frac{1}{c_{k-1}}$ of choosing the bin correctly, given that there are at most c_{k-1} bins with k balls in them even at the end of the game, let alone in some intermediate round. Summing over the different values of k (which each have chance $1/\log(q)$ of being selected), we thus see that B's chance of success is

$$\begin{aligned}\sum_{k=1}^{\log(q)} \frac{1}{\log(q)} \cdot \frac{1}{q^2} \cdot \frac{c_k}{c_{k-1}} &= \frac{1}{q^2}\ \mathsf{ArithmeticMean}\left(\frac{c_1}{c_0}, \dots, \frac{c_{\log(q)}}{c_{\log(q)-1}}\right) \\ &\geq \frac{1}{q^2}\ \mathsf{GeometricMean}\left(\frac{c_1}{c_0}, \dots, \frac{c_{\log(q)}}{c_{\log(q)-1}}\right) \\ &= \frac{1}{q^2}\left(\frac{c_{\log(q)}}{c_0}\right)^{\frac{1}{\log(q)}} \geq \frac{1}{q^2}\left(\frac{1}{q^2}\right)^{\frac{1}{\log(q)}} \\ &= \frac{1}{4q^2}\end{aligned}$$

as claimed, where we used $c_0 = q^2$ and $c_{\log(q)} \geq 1$. □

The above lemma immediately gives us a forger B for Case (2), which succeeds to forge either f_1 or f_2 with probability at least $1/4q^2 > 1/(q\log(q))^2$ for that case. As B chooses randomly which strategy to use and one of the two cases must occur with probability at least $\varepsilon/2$, B's chance of success is at least $\frac{\varepsilon}{2}\min(1/(q\log(q))^2, 1/4q^2) = \varepsilon/2(q\log(q))^2$, completing the WCR proof under our two simplifying assumptions.

General Case. Note that our (main) **Query Order** Assumption (namely that queries to f_3 come before queries to f_1 and f_2) is only used rather weakly, in the sense that A could make its queries in any order as long as the query which completes the collision is a query to f_3. Thus removing this assumption amounts to handling two extra cases, in which collisions are completed with queries to f_1 or f_2 instead of f_3. It turns out these cases can be handled fairly similarly to the f_3 case. The details are deferred to the full version [9].

5 Security of SS-NMAC as a PRF

In this section we show that SS-NMAC is a secure PRF if f is a secure PRF. We will prove a stronger property in Section 6; here we give a proof reducing to the security of encrypted CBC-MAC, which gives a weaker result but a better security bound. The precise statement is the following theorem.

Theorem 3. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ and let H be the SS-NMAC function family. Then, letting $q = \mu/n$ and $\varepsilon = \mathbf{InSec}_f^{\mathrm{prf}}(t, q, qn)$, we have*

$$\mathbf{InSec}_H^{\mathrm{prf}}(t, \tilde{q}, \mu) \le 5q^2/2^n + 4\varepsilon.$$

Proof. Let H^* be the SS-NMAC construction where f_1, f_2, f_3, f_4 are random functions. Then obviously $\mathbf{InSec}_H^{\mathrm{prf}}(t, \tilde{q}, \mu) \le \mathbf{InSec}_{H^*}^{\mathrm{prf}}(t, \tilde{q}, \mu) + 4\varepsilon$, so it suffices to show that $\mathbf{InSec}_{H^*}^{\mathrm{prf}}(t, \tilde{q}, \mu) \le 5q^2/2^n$ where $q = \mu/n$.

We show that $\mathbf{InSec}_{H^*}^{\mathrm{prf}}(t, \tilde{q}, \mu) \le 5q^2/2^n$ by reducing to the security of the "original" encrypted CBC-MAC, which is defined using a function family f of n-bit to n-bit functions by

$$C[f_1, f_2](x_1\|\dots\|x_m) = f_2(\dots f_1(f_1(x_1) \oplus x_2)\dots)$$

Let C^* be the instance of C where f_1, f_2 are random functions (namely, f is the set of all functions $\{0,1\}^n \to \{0,1\}^n$). It is known that $\mathbf{InSec}_{C^*}^{\mathrm{prf}}(t, \tilde{q}, \mu) \le q^2/2^n$ where still $q = \mu/n$ [14]. The security proof in [14] is also easily seen to apply to the case of a three-keyed, "alternating" encrypted CBC-MAC defined by

$$C_A[f_2, f_3, f_4](x_1\|\dots\|x_m) = f_4(\dots f_2(f_3(f_2(x_1) \oplus x_2) \oplus x_3)\dots)$$

in which encryptions by f_2 and f_3 alternate. Thus $\mathbf{InSec}_{C_A^*}^{\mathrm{prf}}(t, \tilde{q}, \mu) \le q^2/2^n$ where C_A^* is the random function implementation of C_A.

Note that C_A becomes H if each block of input is repeated once and encrypted with a call to f_1. Thus a distinguisher D for H^* can be used to obtain a distinguisher D' for C_A^*: sample a key k_1 to simulate the function f_{k_1}, then simulate a query $x_1\|\cdots\|x_m$ of D to the oracle H^* by passing $f_{k_1}(x_1)\|f_{k_1}(x_1)\|f_{k_1}(x_2)$ $\|f_{k_1}(x_2)\cdots\ f_{k_1}(x_m)\|f_{k_1}(x_m)$ to the oracle for C_A^*.

If the oracle is a true instance of C_A^* the answers returned to D look exactly as the answers of an oracle to H^*, so D's chance of distinguishing correctly is unaffected in that case. If on the other hand the oracle is a random function the answers returned to D are independent random values except when the same input is queried twice to the random oracle, which can happen because of collisions in f_{k_1}. The chance of a collision in f_{k_1} when $q = \mu/n$ blocks of message are queried and f_{k_1} is a random function is at most $q^2/2^n$, however, so D and D''s distinguishing advantages differ by at most $q^2/2^n$. Thus, since D' uses twice the message length, we get the desired

$$\mathbf{InSec}^{\mathrm{prf}}_{H^*}(t,\tilde{q},\mu) \leq \mathbf{InSec}^{\mathrm{prf}}_{C_A^*}(t,\tilde{q},2\mu) + q^2/2^n \leq 5q^2/2^n. \qquad \square$$

6 Enhanced PRF Security in the Oracle Cipher Model

In this section, we introduce (following [8]) a strictly *stronger* PRF security notion for block-cipher-based PRFs in the so called *oracle cipher model*, and show that SS-NMAC has (nearly) birthday "oracle cipher security" when instantiated with a secure PRP.

Let H be a function using a fixed-key block cipher f (or a small set of different fixed key block ciphers). Essentially, the oracle cipher model is designed to allow the adversary to view computation transcripts of H, but not including the internals of the block cipher calls. For example, one can imagine that the adversary witnesses a trusted party's computation of H on various inputs, where the trusted party out-sources the block cipher calls to a smart-card, so that the secret keys remain hidden from the adversary. We argue that H is a good random function if, subsequent to viewing a number of such computations, the adversary is unable to distinguish H (queried on new values) from a truly random function.

Let M^f be an oracle Turing machine implementing H. Before the game starts random keys are chosen for the block ciphers, a random function h with same domain and range as H is sampled, and a coin flipped to determine whether the adversary will be in the "real world" or "random world". We allow the adversary two types of queries: "transcript" queries and "oracle" queries. When the adversary A makes a transcript query the transcript of the computation $M^f(x)$ is returned to A. When the adversary makes a oracle query (oracle queries must be distinct from transcript queries), the adversary either gets $H^f(x)$ or $h(x)$ depending on whether it is in the real world or the random world. The adversary wins if it can distinguish the two worlds.

We call the advantage of an adversary at winning this game the *oracle cipher PRF security of* H, denoted $\varepsilon_{\mathrm{oprf}}$. Clearly $\varepsilon_{\mathrm{oprf}} \geq \varepsilon_{\mathrm{prf}}$ for the same number of queries and the same computational resources, since the adversary is free to play the oracle cipher game without making any transcript queries. Let $\mathbf{InSec}^{\mathrm{oprf}}_{H}(t,\tilde{q},\mu)$ be the maximum $\varepsilon_{\mathrm{oprf}}$ over all adversaries running in time at most t, making at most q queries of total (padded) length at most μ, where the running time includes the time necessary to run H and M^f. (Obviously, $\mathbf{InSec}_H^{\mathrm{oprf}}(t, \tilde{q}, \mu)$ implicitly depends on the choice of M.) We have the following theorem showing that the oracle cipher security of SS-NMAC is nearly equivalent to its standard PRF security.

Theorem 4. *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, let $H : \{0,1\}^{4\kappa} \times \mathrm{Dom}(H) \to \{0,1\}^n$ be the SS-NMAC function family, and let M^{f_1,f_2,f_3,f_4} be the natural oracle Turing implementation of SS-NMAC , which makes $3\ell + 1$ oracle calls to compute $H(x)$ on a padded input of ℓ blocks. Then with respect to this oracle Turing machine, and letting $q = \mu/n$ and $\varepsilon = \mathbf{InSec}_f^{\mathrm{prf}}(t, q, qn)$, we have*

$$\mathbf{InSec}_H^{\mathrm{oprf}}(t, \tilde{q}, \mu) \leq 30q^2 \log^2(q)/2^n + 4\varepsilon.$$

Proof. Let H^* be the instantiation of H with a truly random function family instead of with f. We clearly have $\mathbf{InSec}_H^{\mathrm{oprf}}(t, \tilde{q}, \mu) \leq \mathbf{InSec}_{H^*}^{\mathrm{oprf}} + 4\varepsilon$, so it suffices to show $\mathbf{InSec}_{H^*}^{\mathrm{oprf}} \leq 30q^2 \log^2(q)/2^n$.

We now modify the game like so: for each type of query (transcript and oracle), the adversary is allowed to view the transcript of the computation of $H^*(x)$ *up to the application of* f_4. Then for a transcript query the actual application of f_4 is shown as part of the transcript to the adversary, whereas for an oracle query the value of the oracle query is simply appended to the transcript (which will be the value of f_4 in the real world, or else simply the value of the random function h). Note the adversary knows in either case which type of query it is witnessing, but cannot independently verify f_4 for oracle queries unless it happens to make another query later (either transcript or oracle) which results in the same input to f_4. In fact, if the adversary never makes two queries at least one of which is an oracle query that result in the same input to f_4, the two worlds look exactly alike (because f_4 is uniformly random) and the adversary has zero advantage.

Thus the adversary's advantage is upper bounded by its probability of finding a collision at the input to f_4 with free oracle access to f_1, f_2, f_3, which is in turn upper bounded by the collision resistance of the Shrimpton-Stam compression function when instantiated with random functions. Thus Theorem 1 applied with MAC insecurity $1/2^n$ gives $\mathbf{InSec}_{H^*}^{\mathrm{oprf}}(t, \tilde{q}, \mu) \leq 30q^2 \log^2(q)/2^n$, as desired. □

7 Unpredictability vs. Pseudorandomness

Given that our solution is three times slower than CBC-MAC, it is interesting to see if existing block ciphers, such as AES, are indeed more unpredictable than pseudorandom. Notice, even if our n-bit block cipher is completely ideal, it has security $\varepsilon_{\mathrm{prf}} \sim q^2/2^{n+1}$ as a one-block PRF, and a much better security $\varepsilon_{\mathrm{mac}} \sim 1/(2^n - q)$ as a one-block MAC, where q is the number of input queries issued by the attacker. Also, in theory is is trivial to construct artificial block ciphers which are much more unpredictable than pseudorandom. Unfortunately, existing block ciphers are neither ideal nor artificial. For such "real" block ciphers, to the best of our knowledge, this gap between unpredictability and pseudorandomness has not been researched extensively. In part, this might be due to the cryptanalytic "culture" to call the attack truly "successful" if it actually recovers the secret key, which, obviously, will not demonstrate the gap we are seeking here.

We give a (rather weak) example to demonstrate this point. It is well known in complexity theory [20] that no pseudorandom generator with κ-bit key can have security more than $2^{-\kappa/2}$ (against non-uniform attackers), even against *linear tests*.[3] This means that no non-trivial PRF with a κ-bit key can have security $\varepsilon_{\text{prf}} \leq 2^{-\kappa/2}$, even for $q = O(1)$ (e.g., *AES cannot be more than* 2^{-64} *secure, even for* $q = 2$*!*). In contrast, no such limitation is known for unpredictability, even for exponentially high number of queries q (e.g., for all we know, AES might be almost 2^{-128} secure, even for $q = 2^{30}$ or higher). However, the above theoretical "separation" is not considered a "real attack", since the best known way to translate this specific $2^{-\kappa/2}$ distinguishing attack to the key recovery attack takes time $\Omega(2^{\kappa})$, which is trivial.

We hope that our work will motivate further research to understand the gap between unpredictability and pseudorandomness of existing block ciphers, such as AES. In particular, to answer the question if existing modes, such as CBC-MAC or HMAC, should be replaced by slower, but more "resilient" modes, such as SS-NMAC.

References

1. Alon, N., Goldreich, O., Hastad, J., Peralta, R.: Simple Construction of Almost k-wise Independent Random Variables. Random Struct. Algorithms 3(3), 289–304 (1992)
2. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
3. Bellare, M.: New Proofs for NMAC and HMAC: Security without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
4. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
5. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom Functions Re-visited: The Cascade Construction and Its Concrete Security. In: FOCS 1996, pp. 514–523 (1996)
6. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
7. Dodis, Y., Pietrzak, K., Puniya, P.: A New Mode of Operation for Block Ciphers and Length-Preserving MACs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 198–219. Springer, Heidelberg (2008)

[3] Since an ε-secure pseudorandom generator must also be an ε-biased set [12], and such sets must have seed length at least $2\log(1/\varepsilon)$ (see [1]). Thus, $\kappa < 2\log(1/\varepsilon)$.

8. Dodis, Y., Puniya, P.: Feistel Networks Made Public, and Applications. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 534–554. Springer, Heidelberg (2007)
9. Dodis, Y., Steinberger, J.: Message Authentication Codes from Unpredictable Block Ciphers. Full version of this paper, `http://people.csail.mit.edu/dodis/ps/tight-mac.ps`
10. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: STOC 1989, pp. 25–32 (1989)
11. Luby, M., Rackoff, C.: How to construct pseudo-random permutations from pseudo-random functions. SIAM J. Comput. 17(2), 373–386 (1988)
12. Naor, J., Naor, M.: Small-Bias Probability Spaces: Efficient Constructions and Applications. SIAM J. Comput. 22(4), 838–856 (1993)
13. Naor, M., Reingold, O.: From unpredictability to indistinguishability: A simple construction of pseudo-random functions from MACs. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 267–282. Springer, Heidelberg (1998)
14. Petrank, E., Rackoff, C.: CBC MAC for Real-Time Data Sources. J. Cryptology 13(3), 315–338 (2000)
15. Preneel, B., van Oorschot, P.C.: MDx-MAC and Building Fast MACs from Hash Functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
16. Rogaway, P., Steinberger, J.: How to Build a Permutation-Based Hash Function. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008)
17. Shrimpton, T., Stam, M.: Building a Collision-Resistant Compression Function from Non-Compressing Primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008)
18. Simon, D.R.: Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
19. Wang, X., Yu, H.: How to break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
20. Zuckerman, D.: Private communication

How to Encipher Messages on a Small Domain
Deterministic Encryption and the Thorp Shuffle

Ben Morris[1], Phillip Rogaway[2], and Till Stegers[2]

[1] Dept. of Mathematics, University of California, Davis, California 95616, USA
[2] Dept. of Computer Science, University of California, Davis, California 95616, USA

Abstract. We analyze the security of the Thorp shuffle, or, equivalently, a maximally unbalanced Feistel network. Roughly said, the Thorp shuffle on N cards mixes any $N^{1-1/r}$ of them in $O(r \lg N)$ steps. Correspondingly, making $O(r)$ passes of maximally unbalanced Feistel over an n-bit string ensures CCA-security to $2^{n(1-1/r)}$ queries. Our results, which employ Markov-chain techniques, enable the construction of a practical and provably-secure blockcipher-based scheme for deterministically enciphering credit card numbers and the like using a conventional blockcipher.

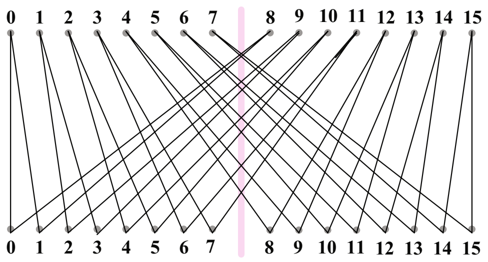
1 Introduction

Small-space encryption. Suppose you want to encrypt a 9-decimal-digit plaintext, say a U.S. social-security number, into a ciphertext that is again a 9-decimal-digit number. A shared key K is used to control the encryption. Syntactically, you seek a cipher E: $\mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{M} = \{0, 1, \ldots, N-1\}$, $N = 10^9$, and $E_K = E(K, \cdot)$ is a permutation for each key $K \in \mathcal{K}$. You aim to construct your scheme from a well-known primitive, say AES, and to prove your scheme is as secure as the primitive from which you start.

The problem is harder than it sounds. You can't just encode each plaintext $M \in \mathcal{M}$ as a 128-bit string and then apply AES, say, as that will return a 128-bit string and projecting back onto $\mathcal{M}$ will destroy permutivity. Standard blockcipher modes of operation are of no use, and constructions like balanced Feistel [17,31] or Benes [1,30] have security that falls off by, at best, the square root of the size of the domain, N. Here N is so small that such a result may provide no practically-useful guarantee.

The above *small-space encryption* problem was first investigated by Black and Rogaway [6], but those authors could find no practical and provably-secure solution for N-values where $q > \sqrt{N}$ queries are feasible but having an encryption take N computational steps is not—values like $2^{20} \leq N \leq 2^{50}$. This paper provides a solution for these troublesome domains.

Thorp shuffle. Our approach is based on the *Thorp shuffle* [40], which works like this. Suppose you want to shuffle N cards, where N is even. Cut the deck into two equal piles. Drop the bottom card from either the left or right pile according to the outcome of a fair coin flip, and then drop the card from the bottom of the other pile. Continue in this way, flipping $N/2$ independent coins



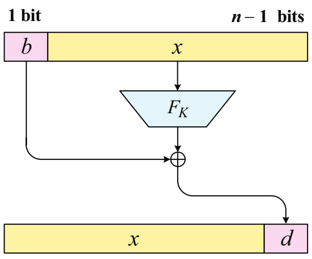


Fig. 1. Two views of the Thorp shuffle (one round). *Left*: each card is paired with the one $N/2$ positions away. For cards at positions x and $x + N/2$, a random bit c (not shown) determines if the cards get mapped to $2x$ and $2x + 1$ or to $2x + 1$ and $2x$. *Right*: each card is regarded as an n-bit string X (assume $N = 2^n$). Now card $b \| x$ gets sent to $x \| b \oplus F_K(x)$ for a uniform (and round-dependent) random function F_K.

and using each to decide if you drop cards left-then-right or right-then-left. This is one *round* of the shuffle; repeat for as many rounds as you like. Expressed a bit more algebraically, for each round $r = 1, 2, \dots, R$ the cards at positions x and $x + N/2$, where $x \in \{0, \dots, N/2 - 1\}$, are moved either to positions $2x$ and $2x + 1$ or else to positions $2x + 1$ and $2x$, which ones being determined by a uniform coin flip $c \in \{0, 1\}$. See the left-hand side of Fig. 1. Let $\mathrm{Th}[N, R]$ denote the Thorp shuffle with message space $\mathcal{M} = \{0, \dots, N - 1\}$ and R rounds.

The potential utility of the Thorp shuffle to cryptography and complexity theory was first noticed by Naor some 20 years ago [27, p. 62], [34, p. 17]. He observed that the Thorp shuffle is *oblivious* in the following sense: one can trace the route of any given card in the deck without attending to the remaining cards in the deck. If the Thorp shuffle mixes cards quickly enough, this property would make it suitable for small-space encryption. Namely, the random bit c used for cards x and $x + N/2$ at round r could be determined by applying a pseudorandom function F, keyed by some underlying key K, to x and r. Conceptually, the string K compactly names all of the $(N/2) \cdot R$ random bits that would be needed to shuffle the entire deck. But because the Thorp shuffle is oblivious, only R of these bits, so that many PRF calls, would be needed to encipher a message x.

Feistel connection. There are a variety of alternative views of what goes on in the Thorp shuffle. The one most resonant to cryptographers is this. Suppose that $N = 2^n$ is a power of two. In this case the Thorp shuffle coincides with a maximally unbalanced Feistel network. In an unbalanced Feistel network [18,36], the left and right portions in the n-bit string that is acted on may have different lengths. Throughout this paper, "maximally unbalanced Feistel" means that the round function takes in $n - 1$ bits and outputs a single bit, a "source-heavy" scheme. See the right-hand side of Fig. 1. A moment's reflection will make clear that, if the round function F_K provides uniform random bits, independently selected for each round, then unbalanced Feistel *is* the Thorp shuffle.

As it takes n rounds of maximally unbalanced Feistel until each bit gets its turn in being replaced, we term n rounds of maximally unbalanced Feistel (or $\lceil \lg N \rceil$ rounds of Thorp) a *pass*. One might hope that the Thorp shuffle mixes the deck well after a small number of passes.

Our results. Assume $N = 2^n$ is a power of two, $r \geq 1$, and let $E = \mathrm{Th}[N, R]$ be the Thorp shuffle with $R = 2rn$ rounds (that is, $2r$ passes). We will show that an adversary mounting a nonadaptive chosen-plaintext attack and making q queries will have advantage that is at most $(q/(r+1)) \cdot (4nq/N)^r$ at distinguishing E from a random permutation on n bits. We prove this bound by regarding the Thorp shuffle of a designated q out of N cards as a Markov chain and applying a coupling argument. To the best of our knowledge, this is the first time that coupling has been used to prove security for a symmetric cryptographic primitive. Using a result of Maurer, Pietrzak, and Renner [21], we can infer that $4r$ passes are enough so that a q-query adversary making an adaptive chosen-ciphertext attack will have advantage at most $(2q/(r+1)) \cdot (4nq/N)^r$ at distinguishing E from a random permutation and its inverse. Put in asymptotic terms, one can construct an n-bit permutation that is CCA-secure to $2^{n(1-1/r)}$ queries by making $4r$ passes of a maximally unbalanced Feistel (its round function being a uniformly random function from $n-1$ bits to 1 bit). This far exceeds what balanced Feistel can achieve, providing a demonstrable separation between the security of balanced and unbalanced Feistel. Finally, we consider a weaker notion of security than customary—withstanding a (nonadaptive) *designated-point attack*. For achieving this, just two passes of unbalanced Feistel are already enough.

In applying the results above to solve the small-space encryption problem using a blockcipher like AES, the number of rounds R becomes the number of blockcipher calls. We describe a trick to reduce this by a factor of five (for a 128-bit blockcipher). We sketch other such "engineering" improvements, like making the constructed cipher tweakable [16], and we tabulate the number of blockcipher calls needed for various provable-security guarantees.

Further related work. Morris proved that the mixing time for the Thorp shuffle—roughly, the number of steps until all $q = N$ cards are ordered nearly uniformly—is polylogarithmic: it is $O(\lg^{44} N)$ [25]. This was subsequently improved to $O(\lg^{19} N)$ [22] and then to $O(\lg^{4} N)$ [23].

Naor and Reingold analyzed unbalanced Feistel constructions, showing, in particular, that one pass over a maximally unbalanced Feistel network that operates on n bits remains secure to nearly $2^{n/2}$ queries.

For *balanced* Feistel, the classical analysis by Luby and Rackoff [17] shows that three rounds provide CPA-security (four rounds for CCA-security) to nearly $2^{n/4}$ queries. This was improved by Maurer and Pietrzak [20], who showed that r rounds of balanced Feistel could withstand about $2^{n/2-1/r}$ queries (in the CCA setting). Patarin [31,29] went on to show that a constant number of rounds (six for CCA-security) was already enough to withstand about $2^{n/2}$ queries. He suggested that enough rounds of maximally unbalanced Feistel ought to achieve security for up to $2^{n(1-\varepsilon)}$ queries [29, p. 527], a conjecture that our work now proves.

Granboulan and Pornin [12] describe a method to perfectly realize a random permutation using a clever shuffling procedure due to Czumaj, Kanarek, Kutyłowski, and Loryś [8]. The shuffle requires one to repeatedly sample in a hypergeometric distribution using parameters that are large and vary during the shuffle. In an implementation, Granboulan and Pornin employ an arbitrary-precision floating-point package to help achieve the needed sampling. In the end, about 10^9 machine cycles are used to encipher on a space of $N < 2^{32}$ points. While improvements may come [41], the method is currently impractical.

Kaplan, Naor, and Reingold describe a method to reduce the number of bits needed to specify a permutation that will appear uniform against some number q of queries [14]. They do this by derandomizing a construction such as the Thorp shuffle. They discuss this case, invoking the result of Morris [25].

Håstad analyzes the mixing time of the following *square lattice shuffle*: given an $m \times m$ array, uniformly permute the entries in each row, and then uniformly permute the entries in each column [13]. He shows that a constant number of such passes are enough to mix well. The shuffle is oblivious, and a recursive realization of it would give rise to another solution to the small-space encryption problem.

The problem of enciphering on a small or unusual-size domain can be regarded as a special case of *format-preserving encryption*, a goal informally described by Brightwell and Smith [7], named by Spies [38], and recently formalized by Bellare and Ristenpart [5] and by Rogaway [33].

In a recent proposal to NIST, Spies [37] describes a blockcipher mode of operation, FFSEM, to encipher on an arbitrary intermediate-size domain $\mathcal{M} = \{0, 1, \ldots, N-1\}$. The mechanism combines the use of a balanced Feistel network and the folklore cycle-walking approach.[1]

THE PROBLEM WITH BALANCED FEISTEL. It seems likely that, for any even n, enough rounds of balanced Feistel using a pseudorandom round function yield a computationally-secure small-domain encryption scheme, even up to $q = 2^n - 2$ queries (recall that a Feistel-determined permutation is always even [28, Th 6.1]). No remotely practical attack is known [28], and the construction is of course quite old. But proofs of security for ciphers made from pseudorandom functions invariably work by proving information-theoretic security and then passing to the complexity-theoretic setting. Since balanced Feistel is information-theoretically *insecure* beyond $2^{n/2}$ queries, such an approach is inherently doomed. More precisely, if the adversary may ask $q = 2^{\theta + n/2}$ queries for some $\theta \geq 0$, then, to have any chance of information-theoretic security, one will need a number of rounds that is at least $r = 2^{\theta+1}$. A simple analysis giving this bound is in the full version of this paper.

[1] Cycle-walking works like this. To construct a cipher E_K that enciphers on $\mathcal{M} = \{0, 1, \ldots, N-1\}$ using a cipher E'_K that works on $\mathcal{M}' = \{0, 1, \ldots, N'-1\}$, where $N' \geq N$, iterate $E'_K(X)$ until the first point is found that lies in $\mathcal{M}$. Return this. The method is efficient if E' is and N' is not too much larger than N.

2 Preliminaries

Ciphers. By a *cipher* we mean a map $E\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ and $\mathcal{M}$ are finite nonempty sets (the *key space* and the *domain*) and $E_K(\cdot) = E(K, \cdot)$ is a permutation on $\mathcal{M}$ for every $K \in \mathcal{K}$. Let $\mathcal{A}$ be an adversary, meaning an algorithm with access to an oracle. For the game used to define E's indistinguishability from a random permutation, the oracle will depend on a permutation $f\colon \mathcal{M} \to \mathcal{M}$. It will respond to a query (enc, x) with $f(x)$ and it will respond to a query (dec, y) with $f^{-1}(y)$. Queries outside of $\{\mathsf{enc}, \mathsf{dec}\} \times \mathcal{M}$ are ignored. Define $\mathbf{Adv}_E^{\mathrm{cca}}(\mathcal{A}) = \mathbf{P}\,(K \overset{\$}{\leftarrow} \mathcal{K} :\ \mathcal{A}^{\pm E_K} \Rightarrow 1) - \mathbf{P}\,(\pi \overset{\$}{\leftarrow} \mathrm{Perm}(\mathcal{M}) :\ \mathcal{A}^{\pm \pi} \Rightarrow 1)$ where $\mathcal{A}^{\pm f}$ denotes $\mathcal{A}$ interacting with the f-dependent oracle just described and $\mathcal{A}^{f} \Rightarrow 1$ is the event that it outputs a 1.

We say that adversary $\mathcal{A}$ is *nonadaptive* if its queries are the same on each and every run. It carries out a *chosen-plaintext* attack if each query is an encryption query, and a *chosen-ciphertext* attack if queries may be either encryption or decryption queries. Let $\mathbf{Adv}_E^{\mathrm{ncpa}}(q) = \max_{\mathcal{A}} \mathbf{Adv}_E^{\mathrm{cca}}(\mathcal{A})$ where the maximum is taken over all nonadaptive adversaries that ask at most q encryption queries and no decryption queries. By the standard averaging argument, the notion is unchanged if nonadaptive adversaries are assumed to be *deterministic*: they statically choose their queries $x_1, \ldots, x_q$. Let $\mathbf{Adv}_E^{\mathrm{cca}}(q) = \max_{\mathcal{A}} \mathbf{Adv}_E^{\mathrm{cca}}(\mathcal{A})$ where the maximum is taken over all adversaries that ask at most q queries.

Markov chains. The next section assumes some familiarity with Markov chains and how to show rapid mixing for them using coupling arguments. See any text on the subject, such as Levin, Peres, and Wilmer [15], for some background on this topic.

Let Ω be a finite nonempty set and let μ, ν be probability distributions on Ω. A *coupling* of μ and ν is a pair of random variables (X, Y), defined on the same probability space, such that the marginal distributions of X and Y are μ and ν, respectively. Let

$$\|\mu - \nu\| = \max_{S \subset \Omega} \mu(S) - \nu(S) = \min_{X \sim \mu,\, Y \sim \nu} \mathbf{P}(X \neq Y) \tag{1}$$

be the total variation distance between μ and ν, where $Z \sim \tau$ means that Z has distribution τ. The minimum in the right-hand side of (1) is taken over all couplings (X, Y) of μ and ν. We shall call a coupling that achieves the minimum an *optimal coupling* of μ and ν.

3 Variational Distance of the Projected Thorp Shuffle

Fix $N = 2^n$. Let $\{\mathrm{Th}_t : t \geq 0\}$ be the Markov chain representing the Thorp shuffle with N cards. More formally, let $\mathcal{C}$ be a set of cardinality N, whose elements we call *cards*. For concreteness, $\mathcal{C} = \{0,1\}^n$. The state space of $\{\mathrm{Th}_t\}$ is the set of bijections from $\mathcal{C}$ to $\{0,1\}^n$. For a card $z \in \mathcal{C}$, we interpret $\mathrm{Th}_t(z)$ as the position of card z at time t.

Let $\mathcal{A}$ be a deterministic adversary that makes exactly q queries. Our proof is based on an analysis of the mixing rate of the Thorp shuffle. However, since $\mathcal{A}$

makes only $q \leq N$ queries, we need only bound the rate at which some q-element subset of the cards mixes. So let $z_1, \ldots, z_q$ be distinct cards in $\mathcal{C}$, and let X_t be the vector of positions of cards $z_1, \ldots, z_q$ at time t. For j in $\{1, \ldots, q\}$ we write $X_t(j)$ for the position of card z_j at time t, and define $X_t(1, \ldots, j) = (X_t(1), \ldots, X_t(j))$.

We shall call X_t the *projected Thorp shuffle.* Note that since the Thorp shuffle is a random walk on a group (see, e.g., [35]), it has uniform stationary distribution. Hence the stationary distribution of X_t, which we denote by π, is uniform over the set of distinct q-tuples of elements from $\{0,1\}^n$. Equivalently, π is the distribution of q samples without replacement from $\{0,1\}^n$. Let τ_t denote the distribution of X_t.

Theorem 1 (Rapid mixing). *Let $N = 2^n$ and $q \in \{1, \ldots, N\}$, $\{X_t \colon t \geq 0\}$ the corresponding projected Thorp shuffle, π its stationary distribution, and τ_t the distribution of X_t. Then, for any $r \geq 1$,*

$$\|\tau_{r(2n-1)} - \pi\| \leq \frac{q}{r+1}\left(\frac{4nq}{N}\right)^r .$$

Proof. For a distribution ν on distinct q-tuples of Ω, define

$$\nu(u_1, \ldots, u_j) = \mathbf{P}\left(Z_1 = u_1, \ldots, Z_j = u_j\right)$$
$$\nu(u_j \mid u_1, \ldots, u_{j-1}) = \mathbf{P}\left(\, Z_j = u_j \mid Z_1 = u_1, \ldots, Z_{j-1} = u_{j-1}\,\right)$$

where $(Z_1, \ldots, Z_q) \sim \nu$. For example, $\tau_t(u_1, \ldots, u_j)$ is the probability that, in the Thorp shuffle, cards $z_1, \ldots, z_j$ land in positions $u_1, \ldots, u_j$ at time t, while $\tau_t(u_j \mid u_1, \ldots, u_{j-1})$ is the probability that at time t card z_j is in position u_j given that cards $z_1, \ldots, z_{j-1}$ are in positions $u_1, \ldots, u_{j-1}$. On the other hand, $\pi(u_j \mid u_1, \ldots u_{j-1})$ is the probability that, in a uniform random ordering, card z_j is in position u_j given that cards $z_1, \ldots, z_{j-1}$ land in positions $u_1, \ldots, u_{j-1}$.

Each of the conditional distributions $\tau_t(\,\cdot \mid u_1, \ldots, u_{j-1})$ converges to uniform as $t \to \infty$. When all of these distributions are "close" to uniform, then τ_t will be close to π. In fact, we only need the conditional distributions to be close "on average," as is formalized in the following lemma, which is proved in Appendix A.

Lemma 2. *Fix a finite nonempty set Ω and let μ and ν be probability distributions supported on q-tuples of elements of Ω, and suppose that $(Z_1, \ldots, Z_q) \sim \mu$. Then*

$$\|\mu - \nu\| \;\leq\; \sum_{l=0}^{q-1} \mathbf{E}\Big(\|\mu(\,\cdot \mid Z_1, \ldots, Z_l) - \nu(\,\cdot \mid Z_1, \ldots, Z_l)\|\Big) . \tag{2}$$

Note that in the above lemma, since $Z_1, \ldots, Z_q$ are random variables (whose joint distribution is governed by μ), so is $\|\mu(\,\cdot \mid Z_1, \ldots, Z_l) - \nu(\,\cdot \mid Z_1, \ldots, Z_l)\|$ for every $l \leq q$; each summand in the right-hand side of (2) is the expectation of one of these random variables.

Coupling arguments. Later in the proof, we will be using a coupling argument to bound $\mathbf{E}\big(\|\mu(\,\cdot \mid Y_1, \ldots, Y_l) - \nu(\,\cdot \mid Y_1, \ldots, Y_l)\|\big)$. Typically, such arguments are used in the following way. There is a Markov chain with transition matrix P and stationary distribution π, started from state x. One wants to estimate the total variation distance $\|P^t(x,\cdot) - \pi\|$ between the distribution of the chain at time t and the stationary distribution. To do so, one constructs a pair process $\{(X_t, Y_t) : t \geq 0\}$, the *coupling*, that satisfies the following conditions:

1. Individually, $\{X_t\}$ and $\{Y_t\}$ are Markov chains with transition matrix P.
2. For every $t \geq 0$, if $X_t = Y_t$ then $X_{t+1} = Y_{t+1}$.
3. We have $X_0 = x$ and $Y_0 \sim \pi$.

The random variable $T = \min\{t : X_t = Y_t\}$ is called the *coupling time.* Note that condition (3) implies that $Y_t \sim \pi$ for all $t \geq 0$. Hence equation (1) implies

$$\begin{aligned} \|P^t(x, \,\cdot\,) - \pi\| &\leq \mathbf{P}\left(X_t \neq Y_t\right) \\ &= \mathbf{P}\left(T > t\right). \end{aligned}$$

The idea is to define the coupling in such a way that T is unlikely to be large.

Defining the coupling. Let τ_t be the distribution of X_t. We wish to use coupling to bound the expected distance between $\tau_t(\,\cdot \mid X_t(1), \ldots, X_t(l))$ and the uniform distribution on $\{0,1\}^n \setminus \{X_t(1), \ldots, X_t(l)\}$, for each $l \in \{1, \ldots, q-1\}$.

Our approach will be as follows. For each value of l we will construct a process $\{U_t\}$ on the same probability space as $\{X_t\}$, to get a coupling $\{(X_t, U_t) \colon t \geq 0\}$. The process $\{U_t\}$ will satisfy the following conditions.

- The positions of the first l cards in U_t always agree with X_t.
- For every t, the distribution of the position of card z_{l+1} at time t, given the positions of cards $z_1, \ldots, z_l$, is uniform.

We begin with a key definition. Say that two cards are *adjacent* at time t if their positions (viewed as elements of $\{0,1\}^n$) are the same except for the first bit (or, viewed as elements of $\{0, \ldots, N-1\}$, they differ by $N/2$).

Let X_t be the projected Thorp shuffle. It will be convenient to use a rule for generating the evolution of X_t that uses q fair coins, $c^1, \ldots, c^q$, each of which is flipped at each step. Formally, each c^j is a sequence $\{c^j_t : t \geq 0\}$ of Bernoulli(1/2) random variables, where we interpret c^j_t as the outcome of coin c^j at time t. We assume that all of the c^j_t are independent.

Note that for a given step, it is enough to describe, for each pair z_i, z_j of adjacent cards, $i < j$, how the position of z_i is updated (since this dictates how the position of z_j must be updated). We shall use the following update rule:

Update rule. For each pair of cards z_i, z_j with $i < j$ that are adjacent at time t, we determine the position of z_i at time $t+1$ using coin c^i and coin flip c^i_t as follows:

1. the first (leftmost) bit of the position of z_i is set to c^i_t, and then
2. the position of z_i undergoes a cyclic left bit shift.

Thus if c_i is at position x at time t then at time $t+1$ it will be at position $2(x \bmod N/2) + c_t^i$, or, in string-oriented notation, at position $x[2..N] \parallel c_t^i$. We claim that if $t \geq n-1$ then for any pair of cards z_i and z_j we have

$$\mathbf{P}\left(z_i \text{ and } z_j \text{ are adjacent at time } t\right) \leq 2^{1-n} \; . \tag{3}$$

To verify this claim, note that (by reordering if necessary) we may assume that $i = 1, j = 2$, and the evolution of X_t is governed by the update rule described above. Let E be the event that z_1 and z_2 are adjacent at time t. In order for E to happen, at each step during times $t-1, \ldots, t-n+1$, when their bits are changed (in step 1 of the update rule), the same change must occur for both z_1 and z_2. Thus $E = A \cap B$, where A is the event that z_1 and z_2 were not adjacent at any of the times $t-1, \ldots, t-n+1$, and B is the event that coins c^i and c^j had the same outcomes at times $t-1, \ldots, t-n+1$. It follows that $\mathbf{P}(E) \leq \mathbf{P}(B) = 2^{1-n}$, and the claim is verified.

We are now ready to describe $\{U_t \colon t \geq 0\}$. The starting state U_0 is constructed as follows.

1. We set $U_0(1, \ldots, l) = X_0(1, \ldots, l)$. That is, cards $z_1, \ldots, z_l$ have the same initial positions in U_0 as X_0.
2. The distribution of $U_0(l+1)$ is uniform over $\{0,1\}^n \setminus \{U_0(1), \ldots, U_0(l)\}$.

(We may assume without loss of generality that the probability space on which $\{X_t \colon t \geq 0\}$ is defined is rich enough to allow us to construct such a U_0.) Note that the final condition implies that for every time t the conditional distribution of $U_t(l)$ given $U_t(1, \ldots, l)$ is uniform over $\{0,1\}^n \setminus \{U_t(1), \ldots, U_t(l)\}$.

We now describe the rule for generating (X_{t+1}, U_{t+1}) from (X_t, U_t). Note that the rule for generating $\{X_t \colon t \geq 0\}$ using coins $c^1, \ldots, c^q$ leads to a natural way to generate the evolution of $\{(X_t, U_t) : t \geq 0\}$. Namely, we use the same coins $c^1, \ldots, c^q$ to update both X_t and U_t in each step. Since the positions of cards $z_1, \ldots, z_l$ initially agree in both X_0 and U_0, and we are using the same coin flips $c_t^1, \ldots, c_t^l$ to update them each step, the positions of these cards remain matched for all times t. Furthermore, note that if at any point the position of card z_{l+1} becomes matched, then it remains matched from then on. Recall that τ_t is the distribution of X_t, and let $Z_1, \ldots, Z_l$ be the positions of cards $z_1, \ldots, z_l$ at time t. (Note that these positions are the same in both X_t and U_t.) By (1), we have

$$\begin{aligned}\|\tau_t(\,\cdot \mid Z_1, \ldots, Z_l) - \pi(\,\cdot \mid Z_1, \ldots, Z_l)\| &\leq \mathbf{P}\left(X_t(l+1) \neq U_t(l+1) \mid Z_1, \ldots, Z_l\right) \\ &= \mathbf{P}\left(T > t \mid Z_1, \ldots, Z_l\right),\end{aligned}$$

where $T = \min\{t : X_t(l+1) = U_t(l+1)\}$ is the coupling time. Taking expectations gives

$$\mathbf{E}\Big(\|\tau_t(\,\cdot \mid Z_1, \ldots, Z_l) - \pi(\,\cdot \mid Z_1, \ldots, Z_l\|\Big) \leq \mathbf{P}\left(T > t\right). \tag{4}$$

We claim that

$$\mathbf{P}\left(T > 2n-1\right) \leq p \, , \tag{5}$$

where $p = nl2^{2-n}$. Let A be the event that at some time in $\{n-1, n, \ldots, 2n-2\}$ card z_{l+1} is adjacent to some card of smaller index in the Y or Z process. Unless A occurs, coupling occurs by time $2n-1$. Summing equation (3) over 2 processes, n timesteps, and l smaller indices verifies the claim by showing that

$$\mathbf{P}(A) \le 2nl \cdot 2^{1-n} = p\,. \tag{6}$$

Note that equation (5) holds regardless of the initial state (X_0, U_0), and that the process $\{(X_t, U_t) : t \ge 0\}$ is itself a Markov chain. Now imagine that we have a sequence of trials where in each trial we run the coupling for an additional $2n-1$ steps. The probability that card z_{l+1} remains unmatched after the first trial is at most p. Furthermore, by the memoryless property of Markov chains, given that card z_{l+1} remained unmatched after the first $r-1$ trials, the conditional probability that it remains unmatched after the r-th trial is again at most p. Hence, by induction, $\mathbf{P}(\text{card } z_{l+1} \text{ remains unmatched after } r \text{ trials}) \le p^r = (nl2^{2-n})^r$, that is,

$$\mathbf{P}(T > r(2n-1)) \le (nl2^{2-n})^r\,. \tag{7}$$

Summing over $l \in \{0, \ldots, q-1\}$ and using Lemma 2 gives

$$\begin{aligned} \|\tau_{r(2n-1)} - \pi\| &\le \sum_{l=0}^{q-1} (nl2^{2-n})^r \le \int_0^q (n2^{2-n})^r x^r \, dx \\ &\le \frac{q^{r+1}}{r+1} \cdot n^r 2^{2r-nr} = \frac{q}{r+1}\left(\frac{4nq}{N}\right)^r\,. \end{aligned}$$ □

4 Pseudorandomness of the Thorp Shuffle

CPA-SECURITY. The total variation distance is identical to the advantage with respect to a (deterministic) nonadaptive chosen-plaintext attack. So, reformulating Theorem 1 in cryptographic terms, what we have shown is the following.

Theorem 3 (nCPA-security, concrete). *Let $N = 2^n$ and $1 \le q \le N$. Then, for any $r \ge 1$,*

$$\mathbf{Adv}^{\mathrm{ncpa}}_{\mathrm{Th}[N, r(2n-1)]}(q) \le \frac{q}{r+1}\left(\frac{4nq}{N}\right)^r\,.$$

TIME REVERSAL. Let $\mathrm{Th}^{-1}[N,R] = (\mathrm{Th}[N,R])^{-1}$ denote the time-reverse Thorp shuffle on N cards with R rounds: in round $r \in \{1, \ldots, R\}$ it sends cards $2x$ and $2x+1$, where $0 \le x < N/2$, either to x and $x + N/2$, or to $x + N/2$ and x, depending on a random bit $c(x,r)$. For N a power of two the forward and reverse Thorp shuffle are "isomorphic" Markov chains in the sense that there is a relabeling of states from $\mathrm{Th}[N,R]$ to $\mathrm{Th}^{-1}[N,R]$ that preserves the transition rule. As a consequence, the bound of Theorem 1 applies to the reverse Thorp shuffle as well, giving us the following.

Corollary 4 (nCPA-security, reverse Thorp). *Let $N = 2^n$ and $1 \le q \le N$. Then, for any $r \ge 1$, $\mathbf{Adv}^{\mathrm{ncpa}}_{\mathrm{Th}^{-1}[N, r(2n-1)]}(q) \le (q/(r+1)) \cdot (4nq/N)^r$.*

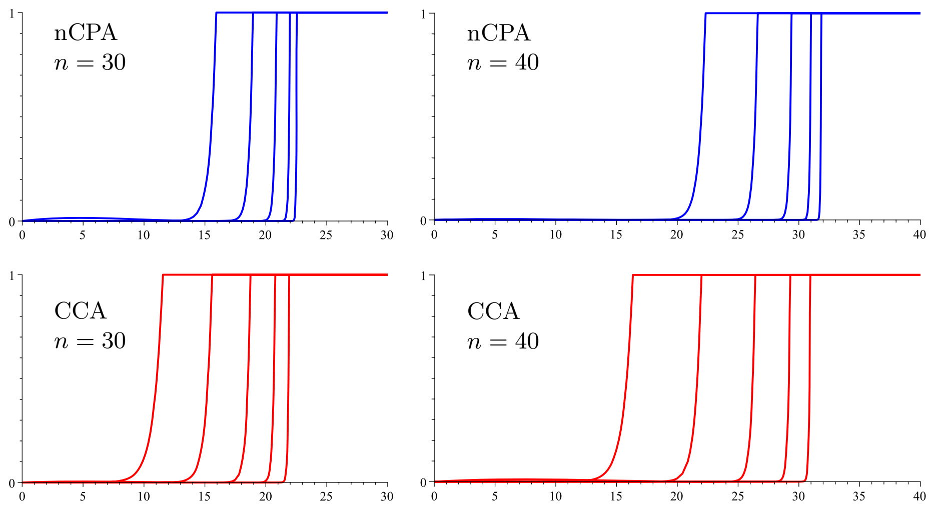


Fig. 2. Proven security of the Thorp shuffle. The x-axis gives the log (base 2) of the number of queries. The y-axis gives an upper bound on an adversary's nCPA advantage by Theorem 3 (top) or its CCA advantage by Theorem 5 (bottom). The curves are for $N = 2^{30}$ points (left) or $N = 2^{40}$ points (right). The curves, from left to right, are for 4, 8, 16, 32, and 64 passes.

CCA-SECURITY. A lovely result of Maurer, Pietrzak, and Renner [21] allows us to easily extend Theorem 3 to a larger class of adversaries, namely, we can trade our nCPA-adversaries for CCA ones. The cost of doing so will be a doubling in the number of rounds, as well as in the advantage bound.

Theorem 5 (CCA-security, concrete). *Let $N = 2^n$ and $1 \le q \le N$. Then, for any $r \ge 1$,*

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathrm{Th}[N,r(4n-2)]}(q) \le \frac{2q}{r+1}\left(\frac{4nq}{N}\right)^r .$$

Proof. We use the second half of Corollary 5 from Maurer et al. [21]. In their notation, we have that $\mathbf{F} = \mathrm{Th}[N, R/2]$, which is stateless, $\mathbf{G} = \mathrm{Th}^{-1}[N, R/2]$, which also stateless, and thus $\Delta_q(\langle \mathbf{F} \triangleright \mathbf{G}^{-1}\rangle, \langle \mathbf{P}\rangle) = \mathbf{Adv}^{\mathrm{cca}}_{\mathrm{Th}[N,R]}(q)$ is bounded above by $\Delta^{\mathsf{NA}}_q(\mathbf{F}, \mathbf{P}) + \Delta^{\mathsf{NA}}_q(\mathbf{G}, \mathbf{P}) = \mathbf{Adv}^{\mathrm{ncpa}}_{\mathrm{Th}[N,R/2]}(q) + \mathbf{Adv}^{\mathrm{ncpa}}_{\mathrm{Th}^{-1}[N,R/2]}(q)$. Note that nonadaptive adversaries in [21] may be probabilistic, but that the best deterministic adversary must do at least as well. Applying Theorem 3 and Corollary 4 to bound the last two summands yields the result. □

GRAPHICAL ILLUSTRATION. The bounds of Theorems 3 and 5 are illustrated in Fig. 4. For example, for 16 passes and $N = 2^{40}$ points (third curve on the bottom right), an adversary must ask at least $2^{26.2}$ queries to have CCA advantage 0.5. For comparison, when applied to a maximally unbalanced Feistel network, the earlier analysis of Naor and Reingold [27, Th 6.2] would have topped out—with

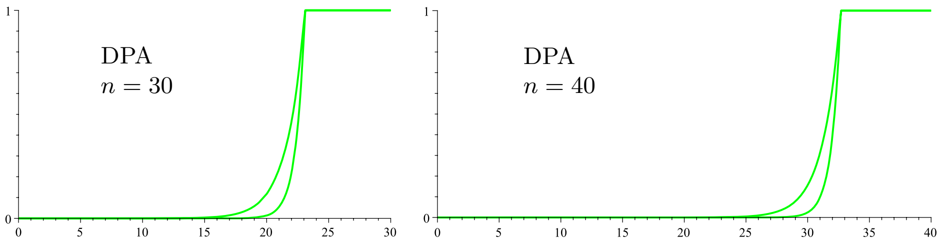


Fig. 3. Proven security of the Thorp shuffle, continued. The x-axis gives the log (base 2) of the number of queries. The y-axis gives an upper bound on an adversary's DPA advantage by Theorem 8, both for $N = 2^{30}$ points (left) and $N = 2^{40}$ points (right). The curves, from left to right, are for two passes and then four.

one pass—at $2^{16.8}$ queries. Had we enciphered strings using a balanced Feistel network instead, then the result of Maurer and Pietrzak [20, Th 1], would give a family of curves (depending, like ours, on how many rounds were performed) that would top out by $2^{18.5}$ queries. Patarin's result for six-round Feistel [31] would apparently be similar, but the concrete security is not explicitly given in that work, and the quantitative bounds are difficult to infer.

Asymptotic interpretation. For an asymptotic interpretation of Theorem 3, fix $r > 0$ and suppose that $q = N^{1-1/r}$ and where, as before, $N = 2^n$. Then

$$\mathbf{Adv}^{\mathrm{ncpa}}_{\mathrm{Th}[N,2rn]}(q) \le \frac{q}{r+1}\left(\frac{4nq}{N}\right)^r = \frac{4^r n^r}{r+1}\cdot\frac{1}{N^{1/r}}\ .$$

In other words, we have upper-bounded the advantage by an expression of the form $(a \log^b N)/N^{1/r}$ for r-dependent constants a and b. Since this goes to 0 as $n \to \infty$, we conclude the following.

Corollary 6 (nCPA-security, asymptotic). *Let $r \ge 1$ be an integer. Then*

$$\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{ncpa}}_{\mathrm{Th}[2^n,2rn]}\left(2^{n(1-1/r)}\right) = 0\ .$$

In English, a maximally-unbalanced Feistel network on n bits employing $2r$ passes maintains security to nearly 2^n queries: specifically, to $2^{n(1-1/r)}$ queries for large enough n. Said differently, you can achieve security up to $N^{1-\varepsilon}$ nonadaptive queries, for any $\varepsilon > 0$, provided you make at least $2 \cdot \lceil 1/\varepsilon \rceil$ passes. This is far better than what a balanced Feistel network can achieve. The asymptotic version of Theorem 5 is similar.

Corollary 7 (CCA-security, asymptotic). *Let $r \ge 1$ be an integer. Then*

$$\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{cca}}_{\mathrm{Th}[2^n,4rn]}\left(2^{n(1-1/r)}\right) = 0\ .$$

Designated-point security. The PRP notion of security formalizes an adversary's inability to detect non-uniform behavior when it sees a *population* of plaintext/ciphertext pairs. Many security notions instead demand that the adversary figure something out about a designated point that it selects: the customary formulations for find-then-guess security, semantic security, unforgeability, and non-malleability are all this way. Weakening the security notion along these lines facilitates a stronger bound for the Thorp shuffle.

Let $E\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be a cipher and let $\mathcal{A}$ be an adversary. We measure the effectiveness of $\mathcal{A}$ in carrying out a *designated-point attack* on E by $\mathbf{Adv}_E^{\mathrm{dpa}}(\mathcal{A}) = \mathbf{P}\left(\mathcal{A}^G \Rightarrow 1\right) - \mathbf{P}\left(\mathcal{A}^H \Rightarrow 1\right)$ where oracles G and H behave like this. Both begin by choosing $K \overset{\$}{\leftarrow} \mathcal{K}$ and then answering queries (enc, x) by $E_K(x)$. Oracle G answers the same way for a query (test, x), but H answers such a query by a uniformly chosen value that has not yet been returned to $\mathcal{A}$. No other types of queries are allowed. The adversary may ask a single test query, its last: once a test query is asked, any subsequent query returns $\perp$. Let $\mathbf{Adv}_E^{\mathrm{dpa}}(q) = \max_{\mathcal{A}} \mathbf{Adv}_E^{\mathrm{dpa}}(\mathcal{A})$ where the maximum is taken over all deterministic nonadaptive adversaries that ask exactly q enc queries. The DPA notion is similar to, but weaker than, the IUP notion investigated by Desai and Miner [9].

Theorem 8 (Designated-point security). *Let $N = 2^n$ and $1 \le q \le N$. Then, for any $r \ge 1$,*

$$\mathbf{Adv}_{\mathrm{Th}[N, r(2n-1)]}^{\mathrm{dpa}}(q) \le \left(\frac{4nq}{N}\right)^r .$$

The proof follows immediately from equations (4) and (7). The bounds are illustrated in Fig. 3. An asymptotic counterpart for the result is as follows.

Corollary 9 (Designated-point security, asymptotic). *For any $\varepsilon > 0$ we have that $\lim_{n\to\infty} \mathbf{Adv}_{\mathrm{Th}[2^n, 2n]}^{\mathrm{dpa}}\left(2^{n(1-\varepsilon)}\right) = 0$.*

More general message spaces. We emphasize that our results on the Thorp shuffle have assumed that the size of the message is a power of two. By using the cycle-walking construction [6], this suffices to encipher messages on any message space $\{0, \ldots, N-1\}$. But the cost of applying this domain transformation can be nearly as bad as an expected doubling in the encryption and decryption time. It would be more desirable for the results to directly apply to Thorp-enciphering for any even N. We expect the full version of this paper to report on such results.

5 Efficiently Realizing the Thorp Shuffle

In this section we sketch a practical realization of Thorp-shuffle encryption. We assume a pseudorandom function $f\colon \mathcal{K} \times \Sigma^* \to \{0,1\}^{128}$. In the analysis, $\rho = f(K, \cdot)$ is regarded as a uniform random function. The translation to the complexity-theoretic setting is standard, the PRF's key K naming a particular ρ. A natural instantiation of f would be the CBC MAC of AES (after a prefix-free encoding of the input [32]). Typically, only one AES call would be needed per PRF invocation.

$p =$	$n = 20$				$n = 30$				$n = 40$			
#passes	#AES	dpa	ncpa	cca	#AES	dpa	ncpa	cca	#AES	dpa	ncpa	cca
2	8	12.7	6.9	—	12	22.1	11.6	—	16	31.7	16.4	—
4	16	13.2	9.4	6.4	24	22.6	15.7	11.2	32	32.2	22.1	15.9
8	32	13.4	11.3	9.1	48	22.8	18.8	15.3	64	32.4	26.5	21.7
16	64	13.6	12.4	11.1	96	23.0	20.8	18.6	128	32.6	29.3	26.3
32	128	13.6	13.4	13.0	192	23.1	22.5	20.0	256	32.6	31.8	30.9
64	256	13.6	13.4	13.0	384	23.1	22.6	21.9	512	32.6	31.8	30.9

Fig. 4. Security and its cost. The columns indicate the domain size $N = 2^n$, the number of passes p, the number of AES calls per encryption (with 5x-speedup), and values $\lg q$ such that our bound on $\mathbf{Adv}^{\mathrm{xxx}}_{\mathrm{Th}[2^n,np]}(q)$ is about 0.5, for xxx $\in$ {dpa, ncpa, cca}.

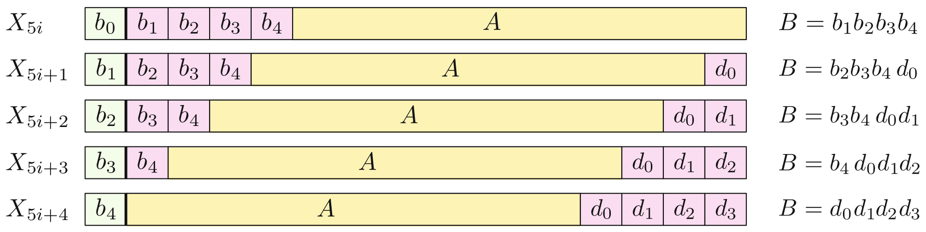
Update rule. Our realization of $\mathrm{Th}[N, R]$ will effectively use a different update rule than that of Section 3: to each $x, x+N/2 \in \{0, \ldots, N-1\}$ and each round r we associate a coin flip $c(x, r)$ that is used to map the card occupying position x in round r to position $2x + c(x, r)$ if $x < N/2$ and to $2x - N + 1 - c$ if $x \geq N/2$. When N is a power of two and $c(x, r) = \rho(x \bmod N/2, r)$, this corresponds to an unbalanced Feistel network.

Five rounds at once. We now describe a technique that lets one compute several rounds of the Thorp shuffle with a single call to the underlying PRF. With a PRF outputting 128 bits, one call will let us do five rounds of enciphering or deciphering instead of just one. We call this the 5x trick. With it, one needs $\lceil R/5 \rceil$ PRF calls to realize $\mathrm{Th}[N, R]$. See Fig. 4 for a representation of how many AES calls would be needed to make various numbers of passes over domains of various sizes, and our proven security bounds in each case.

To explain the idea behind the 5x trick, assume for now that we are enciphering $N = 2^n$ points for some $n \geq 5$. We will use the following notation. If $X \in \{0, 1\}^\ell$ and $i, j \in \{0, \ldots, \ell - 1\}$, we write $X[i]$ for its i-th bit, where the leftmost bit of X is $X[0]$. The substring consisting of the i-th through j-th bit of X is written $X[i..j]$. It is empty if $i > j$. If $v_1, \ldots, v_k$ are bitstrings or integers, $\langle v_1, \ldots, v_k \rangle$ is the tuple $(v_1, \ldots, v_k)$ encoded as a bitstring in some fixed way.

Denote the ciphertext of $X \in \{0, 1\}^n$ after i rounds of Thorp-enciphering by X_i, with $X_0 = X$. Instead of evaluating ρ at $\langle X_i[1 .. n-1], i \rangle$ and using only one of the resulting 128 bits as $c(X_i[1 .. n-1], i)$, we will instead extract a sufficient number of bits to determine all coin flips $c(U, r)$ that may be needed in the same group of five consecutive rounds. Realizing this idea is slightly tricky because it is essential that each coin $c(U, r)$ taken from ρ's output be well-defined no matter how this value may arise, and, at the same time, that it be independent from $c(V, s)$ unless $(U, r) = (V, s)$.

Our strategy is illustrated by Fig. 5. We group the round into runs of five which we call *phases*, beginning with the first. (The last group of five may be shorter.) We exploit the fact that, for $j \in \{0, 1, 2, 3, 4\}$, the strings X_{5i} and X_{5i+j}



X_{5i} b_0 b_1 b_2 b_3 b_4 A $B = b_1b_2b_3b_4$
X_{5i+1} b_1 b_2 b_3 b_4 A d_0 $B = b_2b_3b_4\,d_0$
X_{5i+2} b_2 b_3 b_4 A d_0 d_1 $B = b_3b_4\,d_0d_1$
X_{5i+3} b_3 b_4 A d_0 d_1 d_2 $B = b_4\,d_0d_1d_2$
X_{5i+4} b_4 A d_0 d_1 d_2 d_3 $B = d_0d_1d_2d_3$

Fig. 5. The 5x trick. The lines show successive n-bit strings X_j as we encipher (going down) or decipher (going up) using $\mathrm{Th}[2^n, R]$. For any $A \in \{0,1\}^{n-5}$ and round j divisible by 5, a single call computes the coins associated to all $(n-1)$-bit strings $\star\star\star\star A$ for round j, $\star\star\star A\star$ for round $j+1$, $\star\star A\star\star$ for round $j+2$, $\star A\star\star\star$ for round $j+3$, and $A\star\star\star\star$ for round $j+4$, where each $\star$ may be 0 or 1.

have at least an $(n-5)$-bit substring A in common. We evaluate ρ only on $\langle A, i\rangle$, obtaining 128 random bits. The coin flip $c(X_t[1..n-1], t)$ used to encrypt X_t in round $t = 5i + j$ is then picked out of these 128 bits using $\langle B, j\rangle$ where B is the concatenation of bits $1 \mathinner{..} (4-j)$ and $(n-j+4) \mathinner{..} (n-1)$ of the string X_{5i+j}. The independence requirement is satisfied since the tuple (A, B, i, j) uniquely determines $(X_{5i+j}[1 \mathinner{..} n-1], 5i + j)$.

The real complexity comes when N is *not* a power of 2. Carefully generalizing the idea just sketched by replacing string operations with modulo arithmetic gives rise to the same 5x speedup for Thorp-enciphering any number of points N provided that N is a multiple of 32. We specify this case in Fig. 6. The cycle-walking trick (see footnote 1 on p. 289) can then be used to extend the domain to arbitrary N while never enciphering on a domain that exceeds the size of the original one by more than 31 points. The 5x trick uses $5 \cdot 2^4 = 80$ of the 128 bits output by the PRF. It generalizes to yield a k-fold speedup if the PRF outputs at least $k \cdot 2^{k-1}$ bits, though this necessitates rounding N to the next multiple of 2^k. We omit a proof of the following.

Theorem 10. *Suppose* $32 \mid N$ *and* $R \geq 1$. *If* $\rho\colon \Sigma^* \to \{0,1\}^{128}$ *is a uniform random function then the permutation* Enc_ρ *defined in Fig. 6 realizes* $\mathrm{Th}[N, R]$. *Also,* Dec_ρ *is its inverse. Furthermore, computing* $\mathrm{Enc}_\rho(x)$ *or* $\mathrm{Dec}_\rho(x)$ *requires* ρ *to be evaluated on at most* $\lceil R/5 \rceil$ *points.*

Tweaking. A practical realization for small-space encryption should be *tweakable*, a notion formalized by Liskov, Rivest, and Wagner [16]. The syntax of the cipher is extended to take an additional argument, the *tweak*, and each tweak effectively names a random independent cipher. The algorithm of Fig. 6 is easily modified to accommodate a tweak by adding it to the tuple Y in line 37. As a simple example, an application might need to encipher the upper five digits of a U.S. social security number using a tweak that is the lower four digits.

Variable input length. In Fig. 6, we included the domain size N within the scope of ρ's input (lines 37–38). This makes the scheme secure in the sense of a variable-input-length (VIL) cipher. The property allows to, for example, encipher under a common key database fields that have different domain sizes.

```
10 algorithm $\mathrm{Enc}_\rho(x)$
11 for $r \leftarrow 0$ to $R-1$ do
12     $c \leftarrow F^r_\rho(x \bmod N/2)$
13     if $x < N/2$ then $x \leftarrow 2x + c$
14     else $x \leftarrow 2(x \bmod N/2) + 1 - c$
15 return $x$
```

```
20 algorithm $\mathrm{Dec}_\rho(y)$
21 for $r \leftarrow R-1$ downto $0$ do
22     $c \leftarrow F^r_\rho(y \operatorname{div} 2)$
23     if $c = y \bmod 2$ then $y \leftarrow y \operatorname{div} 2$
24     else $y \leftarrow y \operatorname{div} 2 + N/2$
25 return $x$
```

```
30 algorithm $F^r_\rho(x)$
31 $i \leftarrow r \operatorname{div} 5$
32 $j \leftarrow r \bmod 5$
33 $a \leftarrow (x \operatorname{div} 2^j) \bmod N/32$
34 $hi \leftarrow (x \operatorname{div} 2^j) \operatorname{div} N/32$
35 $lo \leftarrow x \bmod 2^j$
36 $b \leftarrow hi \cdot 2^j + lo$
37 $Y \leftarrow \langle N, i, a \rangle$
38 $table \leftarrow \rho(Y)$
39 $k \leftarrow 16j + b$
40 $c \leftarrow table[k]$
41 return $c$
```

Fig. 6. Realization of $\mathrm{Th}[N, R]$ that incorporates the 5x trick. We assume that $32 \mid N$ and $\rho\colon \{0,1\}^* \to \{0,1\}^{128}$. Line 38 need only be evaluated once every five rounds.

Acknowledgments

Many thanks to Terence Spies, and Voltage Security, for renewing the second author's interest in the small-space encryption problem; without their interest, this work would not exist.

References

1. Aiello, W., Venkatesan, R.: Foiling birthday attacks in length-doubling transformations: Benes: a non-reversible alternative to Feistel. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 307–320. Springer, Heidelberg (1996)
2. Aldous, D., Diaconis, P.: Shuffling cards and stopping times. American Mathematical Monthly 93, 333–348 (1986)
3. Aldous, D., Diaconis, P.: Strong uniform times and finite random walks. Advances in Applied Mathematics 8(1), 69–97 (1987)
4. Bayer, D., Diaconis, P.: Tracing the dovetail shuffle to its lair. Annals of Applied Probability 2(2), 294–313 (1992)
5. Bellare, M., Ristenpart, T.: Format-preserving encryption. Cryptology ePrint report 2009/251
6. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002)
7. Brightwell, M., Smith, H.: Using datatype-preserving encryption to enhance data warehouse security. In: 20th National Information Systems Security Conference Proceedings (NISSC), pp. 141–149 (1997)
8. Czumaj, A., Kanarek, P., Kutyłowski, M., Loryś, K.: Fast generation of random permutations via networks simulation. Algorithmica 21(1) (May 1998)
9. Desai, A., Miner, S.: Concrete security characterizations of PRFs and PRPs: reductions and applications. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 503–516. Springer, Heidelberg (2000)
10. Diaconis, P.: Group representations in Probability and Statistics. Lecture Notes—Monograph series, vol. 11. Institute of Mathematical Statistics (1988)

11. Diaconis, P., Fill, J.: Strong stationary times via a new form of duality. Annals of Probability 18(4), 1483–1522 (1990)
12. Granboulan, L., Pornin, T.: Perfect block ciphers with small blocks. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 452–465. Springer, Heidelberg (2007)
13. Håstad, J.: The square lattice shuffle. Random Structures and Algorithms 29(4), 466–474 (2006)
14. Kaplan, E., Naor, M., Reingold, O.: Derandomized constructions of k-wise (almost) independent permutations. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 354–365. Springer, Heidelberg (2005)
15. Levin, D., Peres, Y., Wilmer, E.: Markov chains and mixing times. American Mathematical Society (2008)
16. Liskov, M., Rivest, R., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
17. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. on Computing 17(2), 373–386 (1988)
18. Lucks, S.: Faster Luby-Rackoff ciphers. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 180–203. Springer, Heidelberg (1996)
19. Maurer, U.: A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 239–255. Springer, Heidelberg (1993)
20. Maurer, U., Pietrzak, K.: The security of many-round Luby-Rackoff pseudo-random permutations. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 544–561. Springer, Heidelberg (2003)
21. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
22. Montenegro, R., Tetali, P.: Mathematical aspects of mixing times in Markov chains. Foundations and Trends in Theoretical Computer Science 1(3) (2006)
23. Morris, B.: Improved mixing time bounds for the Thorp shuffle and L-reversal chain (February 4, 2008) arXiv:0802.0339
24. Morris, B.: The mixing time for simple exclusion. Annals of Applied Probability 16(2) (2006)
25. Morris, B.: The mixing time of the Thorp shuffle. SIAM J. on Computing 38(2), 484–504 (2008); Earlier version in STOC 2005
26. Morris, B., Peres, Y.: Evolving sets, mixing and heat kernel bounds. Probability Theory and Related Fields 133(2), 245–266 (2005)
27. Naor, M., Reingold, O.: On the construction of pseudo-random permutations: Luby-Rackoff revisited. J. of Cryptology 12(1), 29–66 (1999)
28. Patarin, J.: Generic attacks on Feistel schemes. Cryptology ePrint report 2008/036
29. Patarin, J.: Luby-Rackoff: 7 rounds are enough for $2^{n(1-\varepsilon)}$ security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 513–529. Springer, Heidelberg (2003)
30. Patarin, J.: A proof of security in $O(2^n)$ for the Benes scheme. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 209–220. Springer, Heidelberg (2008)
31. Patarin, J.: Security of random Feistel schemes with 5 or more rounds. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 106–122. Springer, Heidelberg (2004)
32. Petrank, E., Rackoff, C.: CBC MAC for real-time data sources. J. of Cryptology 13(3), 315–338 (2000)
33. Rogaway, P.: A synopsis of format-preserving encryption (manuscript) (September 2008)

34. Rudich, S.: Limits on the provable consequences of one-way functions. Ph.D. Thesis, UC Berkeley (1989)
35. Saloff-Coste, L.: Random walks on finite groups. In: Kesten, H. (ed.) Probability on Discrete Structures. Encyclopedia of Mathematical Sciences, vol. 110, pp. 263–346. Springer, Heidelberg (2004)
36. Schneier, B., Kelsey, J.: Unbalanced Feistel networks and block-cipher design. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 121–144. Springer, Heidelberg (1996)
37. Spies, T.: Feistel finite set encryption. NIST submission (February 2008), `http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html`
38. Spies, T.: Personal communications (February 2009)
39. Steiger, W.: A best possible Kolmogoroff-type inequality for martingales and a characteristic property. Annals of Mathematical Statistics 40, 764–769 (1969)
40. Thorp, E.: Nonrandom shuffling with applications to the game of Faro. Journal of the American Statistical Association 68, 842–847 (1973)
41. Zechner, H.: Efficient sampling from continuous and discrete distributions. Ph.D. Thesis, Institute for Statistics, TU Graz (1997)

A Lemma about Total Variation Distance

The following Lemma was used in the proof of Theorem 1. It is essentially Lemma 12 in [24]. We reproduce it here for the convenience of the reader.

Lemma 2. Fix a finite set Ω and let μ and ν be probability distributions supported on q-tuples of elements of Ω, and suppose that $(Z_1, \ldots, Z_q) \sim \mu$. Then

$$\|\mu - \nu\| \le \sum_{l=0}^{q-1} \mathbf{E}\Big(\|\mu(\,\cdot \mid Z_1, \ldots, Z_l) - \nu(\,\cdot \mid Z_1, \ldots, Z_l)\|\Big). \tag{8}$$

Proof. For probability distributions $\widehat{\mu}$ and $\widehat{\nu}$, the total variation distance is

$$\|\widehat{\mu} - \widehat{\nu}\| = \min_{W_1 \sim \widehat{\mu},\ W_2 \sim \widehat{\nu}} \mathbf{P}(W_1 \neq W_2)\,. \tag{9}$$

Thus for every l and $z_1, \ldots, z_l$, we can construct $W_1 \sim \mu(\,\cdot \mid z_1, \ldots, z_l)$ and $W_2 \sim \nu(\,\cdot \mid z_1, \ldots, z_l)$ such that

$$\mathbf{P}(W_1 \neq W_2) = \|\mu(\,\cdot \mid z_1, \ldots, z_l) - \nu(\,\cdot \mid z_1, \ldots, z_l)\|\,.$$

We couple $Z \sim \mu$ with $Y \sim \nu$ as follows. Choose (X_1, Z_1) according to the optimal coupling (i.e., a coupling that achieves the minimum in the RHS of (9)), and subsequently for all l with $1 \le l \le q-1$, if $(Z_1, \ldots, Z_l) = (X_1, \ldots, X_l)$, then choose (Z_{l+1}, X_{l+1}) according to the optimal coupling of $\mu(\,\cdot \mid Z_1, \ldots, Z_l)$ and $\nu(\,\cdot \mid Z_1, \ldots, Z_l)$; else couple (X_{l+1}, Z_{l+1}) in an arbitrary way. Note that

$$\mathbf{P}(Z \neq Y) = \sum_{l=0}^{q-1} \mathbf{P}\Big((Z_1, \ldots, Z_l) = (Y_1, \ldots, Y_l),\ Z_{l+1} \neq Y_{l+1}\Big). \tag{10}$$

But on the event that $(Z_1, \ldots, Z_l) = (Y_1, \ldots, Y_l)$, the pair (Z_{l+1}, Y_{l+1}) is chosen according to the optimal coupling of $\mu(\,\cdot \mid Z_1, \ldots, Z_l)$ and $\nu(\,\cdot \mid Z_1, \ldots, Z_l)$, so the RHS of (10) is at most $\sum_{l=0}^{q-1} \mathbf{E}\Big(\|\mu(\,\cdot \mid Z_1, \ldots, Z_l) - \nu(\,\cdot \mid Z_1, \ldots, Z_l)\|\Big)$. $\square$

How to Hash into Elliptic Curves

Thomas Icart

Sagem Sécurité
Université du Luxembourg
thomas.icart@m4x.org

Abstract. We describe a new explicit function that given an elliptic curve E defined over $\mathbb{F}_{p^n}$, maps elements of $\mathbb{F}_{p^n}$ into E in *deterministic* polynomial time and in a constant number of operations over $\mathbb{F}_{p^n}$. The function requires to compute a cube root. As an application we show how to hash *deterministically* into an elliptic curve.

1 Introduction

Some elliptic curve cryptosystems require to hash into an elliptic curve, for instance the Boneh-Franklin identity based encryption scheme [1]. In this scheme, a particular supersingular elliptic curve is used, for which there exists a one-to-one mapping f from the base field $\mathbb{F}_p$ to the curve. This enables to hash using $f(h(m))$ where h is a classical hash function.

Password-based authentication protocols give another context where hashing into an elliptic curve is sometimes required. For instance, the SPEKE (Simple Password Exponential Key Exchange) [7] and the PAK (Password Authenticated Key exchange) [4] protocols both require a hash algorithm to map the password into a point of the curve. However for ordinary curves the classical approach is inherently probabilistic; this implies that the number of operations required to hash the password may depend on the password itself. As shown in [3] this can lead to a timing attack. Therefore, it would be useful to be able to hash into a curve in a constant number of operations.

The first algorithm mapping $\mathbb{F}_{p^n}$ into an elliptic curve in *deterministic* polynomial time was published by Shallue and Woestijne in ANTS 2006 [10]. The algorithm is based on Skalba's equality [13] and uses a modification of the Tonelli-Shanks algorithm for computing square roots; the algorithm runs in time $\mathcal{O}(\log^4 q)$ for any field size $q = p^n$, and in time $\mathcal{O}(\log^3 q)$ when $q = 3 \mod 4$.

In this paper, we describe another algorithm that given any elliptic curve E defined over $\mathbb{F}_{p^n}$, maps elements of $\mathbb{F}_{p^n}$ into E in deterministic polynomial time, when $p^n = 2 \mod 3$. The new algorithm is based on a rational, explicit function from $\mathbb{F}_{p^n}$ to E, which can be implemented in $\mathcal{O}(\log^3 q)$ time and a constant number of operations over $\mathbb{F}_{p^n}$. Our technique is based on computing a cube root and is simpler than the Shallue and Woestijne algorithm.

As an application we show how to hash *deterministically* and efficiently into an elliptic curve. We provide two different constructions. Our first construction is one-way when the underlying hash function is one-way. The second construction additionally achieves collision resistance when the underlying hash function is collision resistant.

1.1 Related Works

We give a brief description of existing techniques to hash into elliptic curves. An elliptic curve over a field $\mathbb{F}_{p^n}$ where $p > 3$ is defined by a Weierstrass equation:

$$Y^2 = X^3 + aX + b \tag{1}$$

where a and b are elements of $\mathbb{F}_{p^n}$. Throughout this paper, we note $E_{a,b}$ the curve associated to these parameters. It is well known that the set of points forms a group; we denote by $E_{a,b}(\mathbb{F}_{p^n})$ this group and by N its order. We also note $q = p^n$: in particular $\mathbb{F}_q$ is the field of p^n elements.

'Try-and-Increment' Method. The algorithm is described in [2] and works as follows:

```
Input: $u$ an integer.
Output: $Q$, a point of $E_{a,b}(\mathbb{F}_q)$.
1. For $i = 0$ to $k - 1$
   (a) Set $x = u + i$
   (b) If $x^3 + ax + b$ is a quadratic residue in $\mathbb{F}_q$, then return $Q = (x, (x^3 + ax + b)^{1/2})$
2. end For
3. Return $\perp$
```

Heuristically, the algorithm fails to return a point for a fraction 2^{-k} of the inputs, where k is a security parameter. One drawback of the algorithm is that the number of operations is not constant. Indeed the number of steps of the algorithm depends on the input u: approximately half of the u are encoded within 1 step, one fourth within 2 steps, etc. In practice, if the input u has to remain secret, this can lead to a timing attack.

A simple countermeasure consists in outputting the point Q only after the end of the For loop so that the number of steps remains constant. However even with this countermeasure, the running time is not necessarily constant. Namely, if the Legendre symbol is used to determine whether $x^3 + ax + b$ is a quadratic residue, such operation takes $\mathcal{O}(\log^2 q)$ time using quadratic reciprocity laws but in general the number of operations is not constant and depends on the input u. Alternatively, one can compute the Legendre symbol using an exponentiation:

$$\left(\frac{z}{q}\right) = z^{(q-1)/2}.$$

Then the numbers of operations is constant but the running time for computing the Legendre symbol is now $\mathcal{O}(\log^3 q)$.

To summarize, if we do not use any countermeasure then the average running time is $\mathcal{O}(\log^3 q)$ due to the square root computation which takes $\mathcal{O}(\log^3 q)$ when $q = 3 \mod 4$. If we use a constant number of steps k with $k = \mathcal{O}(\log q)$, while computing the Legendre symbol efficiently, the running time is still $k \cdot \mathcal{O}(\log^2 q) + \mathcal{O}(\log^3 q) = \mathcal{O}(\log^3 q)$; however one might still be vulnerable to a timing attack. Finally, if we want to have a constant number of operations, one can use the exponentiation method to compute the Legendre symbol; however the running time becomes $k \cdot \mathcal{O}(\log^3 q) + \mathcal{O}(\log^3 q) = \mathcal{O}(\log^4 q)$. In this paper, we describe an algorithm with running time $\mathcal{O}(\log^3 q)$ and constant number of operations[1].

The 'Twisted' Curves. Another technique consists in using a curve and its twist as suggested in [5]. Given a curve defined by equation (1), one can define the twisted curve of equation

$$cY^2 = X^3 + aX + b$$

where c is a quadratic non-residue in $\mathbb{F}_q$. Then any $x \in \mathbb{F}_q$ is either the abscissa of a point of the original curve or its twist.

One drawback is that a modification of the cryptosystem is required to hide to the adversary which curve is used. In other words, this hashing technique cannot be used as a black box. Another drawback is that it doubles the computation time because the same computations must be performed separately on both curves.

Supersingular Curves. A curve $E_{a,b}$ is called supersingular when $N = q + 1$. When $q \neq 1 \mod 3$, the map $x \mapsto x^3$ is a bijection, therefore the curves of equations

$$Y^2 = X^3 + b$$

are supersingular. One can then define the encoding

$$f : u \mapsto ((u^2 - b)^{1/3}, u) \tag{2}$$

and then the hash function

$$H : m \mapsto ((h(m)^2 - b)^{1/3}, h(m))$$

where h is a classical hash function.

However, the discrete logarithm on these curves is much easier than for ordinary curves. Indeed, such curves have an efficient computable pairing which enables to map the discrete logarithm problem onto a finite field; this is the MOV attack [8]. Therefore in order to avoid this attack, much larger parameters must be used. When no pairing operation is required, it is therefore more efficient to use ordinary curves.

[1] In principle, it should be possible to implement the 'try-and-increment' method in constant time and complexity $\mathcal{O}(\log^3 q)$. For this, one should monitor the running time and eventually use dummy operations. However this could be cumbersome to implement in practice.

The Shallue-Woestijne Algorithm. In ANTS 2006, Andrew Shallue and Christian van de Woestijne have proposed a new algorithm, that generates elliptic curve points in deterministic polynomial time [10].

Let $f(x) = x^3 + ax + b$. The algorithm is based on the Skalba's equality [13]: there exists four maps $X_1(t), X_2(t), X_3(t), X_4(t)$ such that

$$f(X_1(t)) \cdot f(X_2(t)) \cdot f(X_3(t)) = X_4(t)^2.$$

Then in a finite field, for a fixed parameter t, at least one of the $f(X_i(t))$ must be a quadratic residue, which implies that this $X_i(t)$ is an abscissa of a point of the elliptic curve $y^2 = f(x)$.

The computation of $X_1(t), X_2(t), X_3(t), X_4(t)$ and the choice amongst the $X_i(t)$ require to compute square roots in $\mathbb{F}_q$. Computing square roots in $\mathbb{F}_q$ can be done in probabilistic polynomial time using the Tonelli-Shanks algorithm. Thanks to the Skalba equality, the authors of [10] show how to do it deterministically using a modification of the Tonelli-Shanks algorithm, in time $\mathcal{O}(\log^4 q)$. We note that for $q = 3 \mod 4$, computing a square root is simply an exponentiation, which takes $\mathcal{O}(\log^3 q)$. Therefore the Shallue-Woestijne algorithm runs in time $\mathcal{O}(\log^4 q)$ for any field size $q = p^n$, and in time $\mathcal{O}(\log^3 q)$ when $q = 3 \mod 4$.

Using $\boldsymbol{H(m) = h(m).G}$. We note that for most protocols, it is not possible to hash using $H(m) = h(m).G$ where $h(m) \in \mathbb{Z}$ and G is a generator of the group of points of the elliptic curves. Namely in this case, the discrete logarithm of $H(m)$ with respect to G is known, which makes most protocols insecure. For example, it is easy to see that for Boneh-Franklin identity encryption scheme, the attacker can then decrypt any ciphertext. This remains true if we use $H(m) = h_1(m).G_1 + h_2(m).G_2$ or any such linear combination; in this case, the attacker can compute one Boneh-Franklin private key from a set of other private keys by solving a linear system.

2 An Explicit Encoding from $\mathbb{F}_q$ to $E(\mathbb{F}_q)$

We consider the curve $E_{a,b} : Y^2 = X^3 + aX + b$ over the field $\mathbb{F}_{p^n}$ where $p > 3$ and $p^n = 2 \mod 3$. In these finite fields, the function

$$x \mapsto x^3$$

is a bijection with inverse function

$$x \mapsto x^{1/3} = x^{(2p^n-1)/3}.$$

This enables to create a simple parametrization of a subset of the elliptic-curve $E_{a,b}(\mathbb{F}_{p^n})$. To our knowledge, this parametrization is new. Let

$$\begin{aligned} f_{a,b} : \mathbb{F}_{p^n} &\mapsto E_{a,b} \\ u &\mapsto (x, y) \end{aligned}$$

where

$$\begin{aligned} x &= \left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3} \\ y &= ux + v \end{aligned}$$

where

$$v = \frac{3a - u^4}{6u}.$$

For $u = 0$, we fix $f_{a,b}(0) = \mathcal{O}$, the neutral element of the elliptic curve.

Lemma 1. *Let $\mathbb{F}_{p^n}$ be a field where $p^n = 2 \mod 3$ and $p > 3$. For any $u \in \mathbb{F}_{p^n}$, $f_{a,b}(u)$ is a point of $E_{a,b}(\mathbb{F}_{p^n}) : Y^2 = X^3 + aX + b$.*

Proof. For $u \neq 0$, let $(x, y) = f_{a,b}(u)$. From the definition of x:

$$\left(x - \frac{u^2}{3}\right)^3 = v^2 - b - \frac{u^6}{27}.$$

This expands into:

$$x^3 - u^2x^2 + \frac{u^4}{3}x + b - v^2 = 0.$$

Since $u^4/3 = a - 2uv$, this can be rewritten into

$$x^3 - u^2x^2 + (a - 2uv)x + b - v^2 = 0$$

which leads to

$$x^3 + ax + b = u^2x^2 + 2uvx + v^2 = (ux + v)^2$$

and finally $x^3 + ax + b = y^2$. □

We present a similar result in characteristic 2 in appendix A.

Remark 1. We note that if $x \mapsto x^3$ is not a bijection, but $(v^2 - b - u^6/27)$ is a cube in $\mathbb{F}_q$, we can still use the formulas to compute $(x, y) \in E_{a,b}$.

3 Properties of Our New Encoding $f_{a,b}$

Lemma 2. *The function $f_{a,b}$ can be implemented in deterministic polynomial time, with $\mathcal{O}(\log^3 q)$ running time and a constant number of operations over $\mathbb{F}_q$.*

Proof. When $q = 2 \mod 3$, computing $x \mapsto x^{1/3}$ is an exponentiation with exponent $(2q-1)/3$. This can be implemented in a constant number of operations over $\mathbb{F}_q$. We also need to compute $v = (3a - u^4)/6u$, which requires to compute $1/u = u^{q-2}$, which can also be done in a constant number of operations over $\mathbb{F}_q$. The total running time is then $\mathcal{O}(\log^3 q)$. □

In the following, we show how to compute $f_{a,b}^{-1}(P)$ given a point P. This will be used to show the one-wayness and collision resistance properties of the resulting hash function (see Section 4).

Lemma 3. *Let $P = (x, y)$ be a point on the curve $E_{a,b}$. The solutions u_s of $f_{a,b}(u_s) = P$ are the solutions of the polynomial equation:*

$$u^4 - 6u^2x + 6uy - 3a = 0. \tag{3}$$

Proof. The proof is very similar to the proof of Lemma 1. We write $v = \frac{3a-u^4}{6u}$. We show that the two systems are equivalent:

$$\begin{cases} y^2 = x^3 + ax + b \\ u^4 - 6u^2x + 6uy - 3a = 0 \end{cases} \Leftrightarrow \begin{cases} \left(x - \frac{u^2}{3}\right)^3 = v^2 - b - \frac{u^6}{27} \\ y = ux + v \end{cases}$$

From the definition of $f_{a,b}$, this proves the result of the Lemma.

We have:

$$\begin{cases} y^2 = x^3 + ax + b \\ u^4 - 6u^2x + 6uy - 3a = 0 \end{cases} \Leftrightarrow \begin{cases} y^2 = x^3 + ax + b \\ y = ux + v \end{cases}$$

$$\Leftrightarrow \begin{cases} u^2x^2 + 2uvx + v^2 = x^3 + ax + b \\ y = ux + v \end{cases} \Leftrightarrow \begin{cases} x^3 - u^2x^2 + (a - 2uv)x + b - v^2 = 0 \\ y = ux + v \end{cases}$$

$$\Leftrightarrow \begin{cases} x^3 - u^2x^2 + \frac{u^4}{3}x + b - v^2 = 0 \\ y = ux + v \end{cases} \Leftrightarrow \begin{cases} \left(x - \frac{u^2}{3}\right)^3 = v^2 - b - \frac{u^6}{27} \\ y = ux + v \end{cases}$$

□

Lemma 4. *$f_{a,b}^{-1}(P)$ is computable in polynomial time and $\left|f_{a,b}^{-1}(P)\right| \leq 4$, for all $P \in E_{a,b}$,*

Proof. Lemma 3 ensures that to compute $f_{a,b}^{-1}$, it is sufficient to solve a degree 4 equation over $\mathbb{F}_q$. Solving polynomial equations of degree d over a finite field can be solved in $O(d^2 \log^3 q)$ binary operations using the Berlekamp algorithm [12]. For this reason, $f_{a,b}^{-1}$ is computable in polynomial time. Furthermore, since the pre-images are solution of a degree 4 equation over $\mathbb{F}_q$, there are at most 4 solutions for any point P, which implies that $\left|f_{a,b}^{-1}(P)\right| \leq 4$. □

From $\left|f_{a,b}^{-1}(P)\right| \leq 4$ we obtain that our function $f_{a,b}$ generates at least a constant fraction of the elliptic-curve points:

Corollary 1. *Let $E_{a,b}$ be a curve over $\mathbb{F}_q$, where $q = p^n$ with $p > 3$ and $p^n = 2$* mod 3*. We have*

$$\frac{q}{4} \leq |\mathsf{Im}(f_{a,b})| \leq q$$

The bounds for $|\mathsf{Im}(f_{a,b})|$ are not tight. We make the following conjecture:

Conjecture 1. There exists a constant λ such that for any q, a, b

$$\left| |\mathsf{Im}(f_{a,b})| - \frac{5}{8} |E_{a,b}(\mathbb{F}_q)| \right| \leq \lambda\sqrt{q}$$

In the following, we motivate our conjecture. From lemma 3, the size of $\mathsf{Im}(f_{a,b})$ depends on the existence of a solution of the equation $u^4 - 6u^2x + 6uy - 3a = 0$ for a given point (x, y) of $E_{a,b}(\mathbb{F}_q)$. A degree 4 polynomial has no root if and only if it is irreducible or if it is the product of two degree 2 irreducible polynomials. Over any finite field $\mathbb{F}_q$ with large q, it is known that random polynomials of degree d are irreducible with asymptotic probability $1/d$ as q goes to infinity [9]. For this reason, there exist approximately $q^2/2$ irreducible degree 2 monic polynomials. Hence, there exist $\binom{q^2/2}{2} \approx q^4/8$ products of two irreducible degree 2 polynomials. This implies that there exist approximately $q^4/4 + q^4/8 = 3q^4/8$ degree 4 monic polynomials with no root in $\mathbb{F}_q$. For this reason, we can estimate that a fraction 5/8 of random monic degree 4 polynomials have roots. As a consequence the size of $\mathsf{Im}(f_{a,b})$ should be approximately 5/8 of the size of $E_{a,b}$. Our conjecture is made by analogy of the Hasse bound.

Theorem 1 (Hasse Bound). $||E_{a,b}(\mathbb{F}_q)| - q - 1| \leq 2\sqrt{q}$

We have tested our conjecture for all curves $E_{a,b}$ over base field $\mathbb{F}_p$ such that $p = 2 \mod 3$ with $p < 10000$. For all these curves, we have computed the number of points of the curve and we also have computed the number of points in $\mathsf{Im}(f_{a,b})$. After this computation, we found a lower bound for λ as 2.3114.

From this conjecture, we have the following corollary, which gives a deterministic, surjective function onto $E_{a,b}(\mathbb{F}_q)$.

Corollary 2. *If Conjecture 1 is true with $\lambda \leq 3$, if $q = 2 \mod 3$ and $q > 1080$, then:*

$$\begin{aligned} F : (\mathbb{F}_q)^2 &\mapsto E_{a,b}(\mathbb{F}_q) \\ (u_1, u_2) &\mapsto f_{a,b}(u_1) + f_{a,b}(u_2) \end{aligned}$$

is a surjective map.

Proof. To prove that F is surjective, we use the drawer principle. Given a point $P \in E_{a,b}(\mathbb{F}_q)$, the set $S_1 = \{P - f_{a,b}(u)\}_{u \in \mathbb{F}_q}$ is made of at least $5q/8 - 3\sqrt{q + 1 + 2\sqrt{q}}$ points. The set $S_2 = \{f_{a,b}(u)\}_{u \in \mathbb{F}_q}$ is also made of the same number of points. This implies that the set $S_1 \cap S_2$ is not empty if $|S_1| + |S_2| > |E_{a,b}(\mathbb{F}_q)|$. This is always true when

$$2\left(\frac{5q}{8} - 3\sqrt{q + 1 + 2\sqrt{q}}\right) > q + 1 + 2\sqrt{q}$$

which leads to $q > 1080$. □

Finally, we note that computing discrete logarithms of $f_{a,b}(u)$ is hard if computing discrete logarithms in $E_{a,b}(\mathbb{F}_q)$ is hard. This is because the function $f_{a,b}$ is efficiently invertible and $\left|f_{a,b}^{-1}(P)\right| \leq 4$ for any P. Let G be a generator of $E_{a,b}(\mathbb{F}_q)$. If we are given as input a random point P, with probability at least $1/4$ we have that $P \in \mathsf{Im}(f_{a,b})$, so we can compute $u \in \mathbb{F}_q$ such that $P = f_{a,b}(u)$. Then if an algorithm can compute x such that $f_{a,b}(u) = x.G$, this gives x such that $P = x.G$. This shows that if an algorithm can compute the discrete logarithm of $f_{a,b}(u)$, then such algorithm can be used to compute the discrete logarithm in $E_{a,b}(\mathbb{F}_q)$. The same argument applies to any encoding function f which is polynomially invertible on its outputs and with a polynomially bounded pre-image size. The argument can be easily extended to show that for any generator base $(G_1, ..., G_n)$, computing $x_1, \ldots, x_n$ such that $f_{a,b}(x) = \sum_i x_i.G_i$ is hard if computing discrete logarithms in $E_{a,b}(\mathbb{F}_q)$ is hard.

4 How to Hash onto Elliptic Curves

Given a function f into an elliptic curve E, we describe two constructions of hash functions into E. We define L as the maximal size of $f^{-1}(P)$ where P is any point on E:

$$L = \max_{P \in E}(\left|f^{-1}(P)\right|)$$

For our encoding function $f_{a,b}$, we have $L \leq 4$ (see lemma 4). We note that if we work in a subgroup of E of order n with cofactor r, we can use the encoding function $f'_{a,b} = r.f_{a,b}$. If r is relatively prime to n, then we must have $L \leq 4r$.

Our first construction is as follows: given a hash function $h : \{0,1\}^* \mapsto \mathbb{F}_q$, we define

$$H(m) = f(h(m))$$

as a hash function into the curve $E_{a,b}(\mathbb{F}_q)$. In the following, we show that H is one-way if h is one way.

4.1 One-Wayness

Definition 1. *A hash function is (t, ε)-one-way, if any algorithm running in time t, when given a random $y \in \mathsf{Im}(h)$ as input, outputs m such that $h(m) = y$ with probability at most ε. A hash function is one-way if ε is negligible for any polynomial t in the security parameter.*

Lemma 5. *If h is a (t, ε)-one-way hash function then H is (t', ε')-one-way where $\varepsilon' = L^2\varepsilon$, where $L = \max_{P \in E}(\left|f^{-1}(P)\right|)$. Therefore, if L is polynomial in the security parameter and h is one-way, then H is one-way.*

The proof is done in the full version of this paper [6].

4.2 Collision Resistance

Definition 2. *A family $\mathcal{H}$ of hash functions is (t, ε)-collision-resistant, if any algorithm running in time t, when given a random $h \in \mathcal{H}$, outputs (m, m') such that $h(m) = h(m')$ with probability at most ε.*

Our first construction is easily extended to hash function families: given a family $\mathcal{H}$ of hash functions, we define for each $h \in \mathcal{H}$ the function $H = f \circ h$. We then study whether the family of hash functions formed by the H is collision resistant.

A collision to one H occurs if and only if:

1. there exists m and m' such that $h(m) = h(m')$; this is a collision for h,
2. or $f(u) = f(u')$ for $u = h(m), u' = h(m')$ and $u \neq u'$; this is a collision for f.

In the following, we argue that we cannot prove the collision resistance of H based on the collision resistance of h only. Namely, we note that given a hash function h, it is easy to construct an elliptic curve with collisions on $H = f_{a,b} \circ h$. Indeed, given (m, m'), let $u = h(m)$ and $u' = h(m')$. From this couple (u, u'), we compute the degree 4 polynomial:

$$(X - u)(X - u')(X^2 + (u + u')X - w) \tag{4}$$

where w is a randomly chosen element in $\mathbb{F}_q$. This polynomial is equal to:

$$X^4 - 6xX^2 + 6yX - 3a$$

where

$$x = -\frac{uu' + w - (u + u')^2}{6}, \quad y = \frac{(u + u')(uu' - w)}{6}, \quad a = -\frac{uu'w}{3}.$$

Let $b = y^2 - x^3 - ax$. Hence (x, y) is a point on the elliptic curve $E_{a,b}$ by definition of b. For this reason, a preimage of (x, y) through $f_{a,b}$ is a solution of the equation:

$$X^4 - 6xX^2 + 6yX - 3a = 0 \tag{5}$$

which is exactly the polynomial (4) by definition of x, y and a. For this reason, u and u' are solutions of the equations (4) and are preimages of (x, y). Hence (m, m') is a collision for $H = f_{a,b} \circ h$.

However, if $E_{a,b}$ is defined independently from h, it seems difficult to find (m, m') such that $f_{a,b}(y) = f_{a,b}(y')$ where $y = h(m)$ and $y' = h(m')$. In this case, H should be collision resistant. We cannot prove that H is collision resistant based only on the collision resistance of h, and we clearly need some additional properties on h. In the next section, we provide a different construction for which collision resistance can be proved based only on the collision resistance of h.

4.3 Making f Collision Free

In this section, we show how to construct a family $\mathcal{G}$ of functions derived from an encoding function f, which is collision free except with negligible probability. Then given a hash function h and given $g \in \mathcal{G}$, $H'(m) = g(h(m))$ will be collision resistant assuming that h is collision resistant.

Definition 3. *A family $\mathcal{G}$ of functions is ε-collision-free if the probability that $g \in \mathcal{G}$ has a collision is at most ε.*

In other words, a collision-free family of functions is a family in which most functions are injective. To construct such collision free family, we use the notion of family of pair-wise independent functions.

Definition 4 (Pair-wise Independence). *A family $\mathcal{V}$ of functions $v : R \mapsto S$ is ε-pair-wise independent if given any couple $(r_1, r_2) \in R^2$ with $r_1 \neq r_2$ and any couple $(s_1, s_2) \in S^2$:*

$$\Pr_{v \in \mathcal{V}} [v(r_1) = s_1 \wedge v(r_2) = s_2] \leq \varepsilon.$$

The following theorem shows that the family $\mathcal{G} = \{f \circ v\}_{v \in \mathcal{V}}$ is collision free. The proof is in the full version of the paper [6].

Theorem 2. *Let $f : S \mapsto T$ be a function such that $\left|f^{-1}(t)\right| \leq L$ for all $t \in T$. Let $\mathcal{V}$ be a family of ε-pair-wise independent functions from R to S. Then the family $\mathcal{G} = (f \circ v)_{v \in \mathcal{V}}$ is ε'-collision-free where*

$$\varepsilon' = |R|^2 \cdot |S| \cdot L \cdot \varepsilon.$$

Therefore, our second construction is as follows. Given a security parameter k and an integer $q = p^n$ with $q \geq 2^k$, we consider the following family of functions:

$$\begin{aligned} (v_{c,d})_{c,d \in \mathbb{F}_q} : \{0,1\}^k &\mapsto \mathbb{F}_q \\ x &\mapsto c \cdot x + d \end{aligned}$$

where x is seen as an element in $\mathbb{F}_q$. It is easy to see that this family is $1/q^2$-pair-wise independent.

Given an elliptic curve E, we combine the encoding function f with the functions in the $v_{c,d}$ family to get a collision-free family $\mathcal{G}$:

$$\begin{aligned} \mathcal{G} = (f \circ v_{c,d})_{c,d \in \mathbb{F}_q} : \{0,1\}^k &\mapsto E(\mathbb{F}_q) \\ x &\mapsto f(c \cdot x + d) \end{aligned}$$

Finally, given a family $\mathcal{H}$ of collision-resistant functions $h : \{0,1\}^* \mapsto \{0,1\}^k$, we construct the following family of hash functions into the curve E:

$$\begin{aligned} \mathcal{H}_E = (f \circ v_{c,d} \circ h)_{\substack{c,\, d \,\in\, \mathbb{F}_q \\ h \,\in\, \mathcal{H}}} : \{0,1\}^* &\mapsto E(\mathbb{F}_q) \\ m &\mapsto f(c \cdot h(m) + d) \end{aligned}$$

Theorem 3. *If $\mathcal{H}$ is a (t, ε)-collision resistant family of hash functions, then $\mathcal{H}_E$ is a (t', ε')-collision resistant family of hash functions where*

$$\varepsilon' = \varepsilon + L\frac{2^{2k}}{q}.$$

The proof is in the full version of the paper [6].

Note that if we take q of size $5k/2$ bits, we obtain $\varepsilon' \leq \varepsilon + 2^{-k/2}$. Therefore if we have a family $\mathcal{H}$ of k-bit ε-collision resistant hash functions where $\varepsilon = 2^{-k/2}$, we obtain the same security level for $\mathcal{H}_E$, namely $\varepsilon' = 2^{-k/2+1}$.

In practice, given a curve E defined modulo a prime p, we randomly select $c, d \in \mathbb{F}_p$ and a function $h \in \mathcal{H}$; this defines a hash function:

$$\begin{aligned} H_E : \{0,1\}^* &\mapsto E(\mathbb{F}_p) \\ m &\mapsto f(c \cdot h(m) + d) \end{aligned}$$

Finally, we have that H_E is a one way hash function when $c \neq 0$:

Lemma 6. *If h is a (t, ε)-one-way hash function, then for any c, d with $c \neq 0$, $H_E = f \circ v_{c,d} \circ h$ is a (t', ε')-one way hash function ,with $\varepsilon' = L^2 \cdot \varepsilon$.*

The proof is in the full version of the paper [6].

We note that this second construction requires a much larger q than the previous construction. For example, for a 160-bit hash function h, the first construction requires a 160-bit integer q, whereas our second construction requires q to be of size $5 \cdot 160/2 = 400$ bits.

5 Practical Implementations

In this section we compare the running time needed by various encodings into elliptic-curves. We first consider our function $f_{a,b}$ with Euclide's algorithm to implement the inversion in $\mathbb{F}_p$; we also consider $f_{a,b}$ (v2) with an exponentiation instead in order to have a constant number of operations over $\mathbb{F}_p$.

We have also implemented the various 'try-and-increment' algorithms: the classic algorithm, the algorithm with a constant number of steps but with fast Legendre symbol (v2) and the algorithm with a constant number of operations using an exponentiation for the Legendre symbol (v3). We have also implemented the encoding defined by equation (2) for a supersingular elliptic-curve; in this case we have used a finite field ensuring the same security level as for ordinary elliptic curves.

The implementation has been done on 10 different 160-bit primes, randomly chosen such that $p = 2 \mod 3$ and $p = 3 \mod 4$. For every prime, 10 different couples of parameters (a, b) have been randomly chosen. And on these 10 different curves, we runned every algorithm 1000 times on random inputs. We used a 512-bit prime for the supersingular curves case.

We obtain that when a constant running time is required, our method performs much better than the 'try-and-increment' algorithm and the algorithm for supersingular curves. It also performs slightly better even when a constant running time is not required.

Table 1. Average Time of each Algorithms using the Number Theory Library (NTL) [11] and running on a laptop using the Intel®Core$^{\text{TM}}$2 Duo T7100 chip at a frequency of 1, 80 Ghz

Algorithm	Constant Running Time	Running Time
$f_{a,b}$	No	0.22 ms
Try and Increment	No	0.24 ms
Try and Increment v2	No	1.86 ms
$f_{a,b}$ v2	Yes	0.40 ms
Try and Increment v3	Yes	16.10 ms
Supersingular Curves	Yes	3.67 ms

6 Conclusion

We have provided a new algorithm that encodes an integer into an elliptic curve point in a constant number of field operations. This encoding exists for any curve under the condition that the map $x \mapsto x^3$ is a bijection on the base field. This encoding is efficiently computable with the same complexity as one exponentiation and one inversion on the base field.

From our encoding, we have defined two constructions, which enable to hash into an elliptic curve. The first construction is provably one-way and the second is provably one-way and collision resistant in the standard model. Our algorithm can be used for password based authentication protocol over elliptic curves. Indeed, it enables to efficiently encode passwords or PIN-codes into points of the curve in a constant number of field operations.

We also note that our encoding enables to compute points of elliptic curves over RSA rings without knowing the factorization of $N = pq$. Consider the following problem: given $N = pq$ where p and q are prime integers and a, b in $\mathbb{Z}_N$, find (x, y) such that $y^2 = x^3 + ax + b \mod N$. Previously, factoring N was required to compute such (x, y). Our function $f_{a,b}$ proves that a cube root oracle is actually sufficient.

Acknowledgments. I wish to thank Jean-Sébastien Coron for the time he spent to help me write this paper. I also thank Julien Bringer, Hervé Chabanne, Bruno Kindarji and the anonymous referees for their helpful comments.

References

1. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
2. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptology 17(4), 297–319 (2004)

3. Boyd, C., Montague, P., Nguyen, K.Q.: Elliptic curve based password authenticated key exchange protocols. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 487–501. Springer, Heidelberg (2001)
4. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
5. Chevassut, O., Fouque, P.-A., Gaudry, P., Pointcheval, D.: The twist-augmented technique for key exchange. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 410–426. Springer, Heidelberg (2006)
6. Icart, T.: How to hash into an elliptic-curve. Publicly, `http://eprint.iacr.org/2009/226`
7. Jablon, D.P.: Strong password-only authenticated key exchange. SIGCOMM Comput. Commun. Rev. 26(5), 5–26 (1996)
8. Menezes, A., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
9. Sedgewick, R., Flajolet, P.: An Introduction to the Analysis of Algorithms, 512 pages. Addison-Wesley Publishing Company, Reading (1996)
10. Shallue, A., van de Woestijne, C.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 510–524. Springer, Heidelberg (2006)
11. Shoup, V.: Ntl, Number Theory C++ Library, `http://www.shoup.net/ntl/`
12. Shoup, V.: A new polynomial factorization algorithm and its implementation. J. Symb. Comput. 20(4), 363–397 (1995)
13. Skalba, M.: Points on elliptic curves over finite fields. Acta Arith. 117, 293–301 (2005)

A An Explicit Encoding from $\mathbb{F}_{2^n}$ to $E(\mathbb{F}_{2^n})$

The equations which define elliptic curves in characteristic 2 are somehow different from the Weierstrass equation:

$$Y^2 + XY = X^3 + aX^2 + b$$

where a and b are elements of $\mathbb{F}_{2^n}$. For an odd n, the map $x \mapsto x^3$ is a bijection. Let

$$\begin{aligned} f_{a,b} : \mathbb{F}_{2^n} &\mapsto (\mathbb{F}_{2^n})^2 \\ u &\mapsto (x, ux + v^2) \end{aligned}$$

where

$$\begin{aligned} v &= a + u + u^2 \\ x &= (v^4 + v^3 + b)^{1/3} + v. \end{aligned}$$

Lemma 7. *Let $\mathbb{F}_{2^n}$ be a field with n odd. For any $u \in \mathbb{F}_{2^n}$, $f_{a,b}(u)$ is a point of $E_{a,b} : Y^2 + XY = X^3 + aX^2 + b$.*

Proof. Given a parameter u, let (x, y) be $f_{a,b}(u)$. We have the following equations for x, u and v:

$$\begin{aligned} 0 &= (x + v)^3 + b + v^3 + v^4 \\ &= x^3 + vx^2 + v^2x + b + v^4. \end{aligned}$$

Since $v = a + u + u^2$, this can be rewritten into:

$$\begin{aligned} x^3 + ax^2 + b &= ux^2 + u^2x^2 + v^2x + v^4 \\ &= (ux + v^2)((u + 1)x + v^2) = y(x + y) \end{aligned}$$

Hence, $(x, y) = f_{a,b}(u)$ is a point of $E_{a,b}$. □

A.1 Cardinality of $\mathsf{Im}(f_{a,b})$ in Characteristic 2

As in the case of the characteristic p, it is possible to bound the $|\mathsf{Im}(f_{a,b})|$.

Theorem 4. $2^{n-2} < |\mathsf{Im}(f_{a,b})| \leq 2^n$

Proof. The inequality $|\mathsf{Im}(f_{a,b})| \leq 2^n$ is the consequence that $f_{a,b}$ is a function.

The other side of the inequality $2^{n-2} < |\mathsf{Im}(f_{a,b})|$ can be explained thanks to the equation $y = ux + a^2 + u^2 + u^4$. As for the characteristic p, the second equation of the Lemma 7 is enough to inverse $f_{a,b}$. This equation can be rewritten as

$$0 = y + a + ux + u^2 + u^4$$

Given a point (x, y), if u is a solution of this equation then $f_{a,b}(u) = (x, y)$. Since the equation is of degree 4, there are at most 4 different u for each point image of $f_{a,b}$. For this reason, there are at least $2^n/4 = 2^{n-2}$ points in $\mathsf{Im}(f_{a,b})$. □

Batch Binary Edwards

Daniel J. Bernstein[⋆]

Department of Computer Science (MC 152)
The University of Illinois at Chicago
Chicago, IL 60607–7053
djb@cr.yp.to

Abstract. This paper sets new software speed records for high-security Diffie-Hellman computations, specifically 251-bit elliptic-curve variable-base-point scalar multiplication. In one second of computation on a $200 Core 2 Quad Q6600 CPU, this paper's software performs 30000 251-bit scalar multiplications on the binary Edwards curve $d(x+x^2+y+y^2) = (x+x^2)(y+y^2)$ over the field $\mathbf{F}_2[t]/(t^{251}+t^7+t^4+t^2+1)$ where $d = t^{57}+t^{54}+t^{44}+1$. The paper's field-arithmetic techniques can be applied in much more generality but have a particularly efficient interaction with the completeness of addition formulas for binary Edwards curves.

Keywords: Scalar multiplication, Diffie–Hellman, batch throughput, vectorization, Karatsuba, Toom, elliptic curves, binary Edwards curves, differential addition, complete addition formulas.

1 Introduction

Which curves should one choose for elliptic-curve cryptography?

The first and most fundamental choice is between curves defined over binary (i.e., characteristic-2) finite fields and curves defined over non-binary fields. For example:

- NIST's standard K-283 curve is the subfield curve $y^2 + xy = x^3 + 1$ over the field $\mathbf{F}_2[t]/(t^{283}+t^{12}+t^7+t^5+1)$. NIST's standard B-283 curve is a particular non-subfield curve over the same binary field.
- NIST's standard P-256 curve is a particular curve over the prime field $\mathbf{Z}/(2^{256}-2^{224}+2^{192}+2^{96}-1)$.

Multiplication in the polynomial ring $\mathbf{F}_2[t]$ is, at first glance, just like multiplication in $\mathbf{Z}$ but skips all the carries. Furthermore, squaring in $\mathbf{F}_2[t]$ is simply a relabeling of exponents. One might therefore guess that curves over binary fields are considerably faster than curves over large-characteristic fields.

[⋆] Permanent ID of this document: 4d7766189e82c1381774dc840d05267b. Date of this document: 2009.06.03. This work was supported by the National Science Foundation under grant ITR–0716498.

However, the software speed records for elliptic-curve cryptography are—and for many years have been—held by large-characteristic fields. The fastest Diffie–Hellman speeds (i.e., speeds for variable-base-point scalar multiplication $n, P \mapsto nP$) reported in ECRYPT's publicly verifiable benchmarks [15] on a single core of an Intel Core 2 Quad Q6600 (6fb, `utrecht`) are

- 321012 cycles for field size $(2^{127}-1)^2$ (using software from Galbraith, Lin, and Scott, combining Edwards curves with the idea of [33]),
- 386739 cycles for field size $2^{255}-19$ (using software from Gaudry and Thomé announced in [35]), and
- 855036 cycles for field size 2^{251} (also using software from [35]).

Similar comments apply to older processors: for example, Fong, Hankerson, López, and Menezes in [30, Table 6] report 1720000 cycles on a Pentium III for field size 2^{233}, while [12] reports 832457 cycles on a Pentium III for field size $2^{255}-19$. Subfield curves provide some speedups in the binary case—for example, Hankerson et al. in [38, Table 7] report 1740000 cycles on a Pentium II for field size 2^{283}, and Hankerson, Karabina, and Menezes in [39, Table 5] report 758000 cycles on a Xeon 5460 (similar to a Core 2 Quad) for field size 2^{254}—but binary fields seem to have no hope of catching up to large-characteristic fields.

Why are large-characteristic fields so much faster than binary fields? The conventional explanation is that today's popular CPUs include fast "integer-multiplication" (and "floating-point multiplication") instructions that multiply medium-size elements of $\mathbf{Z}$, but do not include instructions to multiply medium-size elements of $\mathbf{F}_2[t]$. Of course, one can multiply in $\mathbf{F}_2[t]$ by combining simpler CPU instructions, but the multiplication instructions for $\mathbf{Z}$ are much faster, outweighing any possible advantages of characteristic 2. This effect has been intensified by the transition from 32-bit processors to 64-bit processors: 64-bit multipliers are even more powerful than 32-bit multipliers.

Why have CPU designers decided to include a circuit for multiplication in $\mathbf{Z}$ and not a smaller, faster circuit for multiplication in $\mathbf{F}_2[t]$? The conventional explanation is as follows. Most of the computer users who care about CPU speed are measuring performance of weather simulation, movie decompression, video games, etc. These applications rely heavily on multiplication in $\mathbf{Z}$, rewarding CPUs that include integer multipliers, floating-point multipliers, etc. The same applications make very little use of multiplication in $\mathbf{F}_2[t]$.

New speed records. This paper introduces new software named `BBE251` for scalar multiplication on a high-security binary elliptic curve, specifically the binary Edwards curve $d(x+x^2+y+y^2) = (x+x^2)(y+y^2)$ over k, where $k = \mathbf{F}_{2^{251}} = \mathbf{F}_2[t]/(t^{251}+t^7+t^4+t^2+1)$ and $d = t^{57}+t^{54}+t^{44}+1 \in k$. This curve has group order $4 \cdot \text{prime}$ and twist order $2 \cdot \text{prime}$, and it satisfies all the usual elliptic-curve security criteria; see Section 3.

`BBE251` is so fast that it sets new speed records not just for binary elliptic curves, but for *all* elliptic curves. For example, a benchmark of a batch of 1024 independent scalar multiplications took 321866514 cycles on a single core of a Core 2 Quad Q6600 (6fb)—a cost of just 314323 Core 2 cycles per scalar multiplication, improving upon all previous results. The Sage computer-algebra system [63] was used to check a random sampling of outputs.

Readers and potential users are cautioned that BBE251 does *not* compute one scalar multiplication in 314323 cycles. The software is given a *batch* of curve points $P_1, P_2, P_3, \ldots$ and an equal-length batch of integers $n_1, n_2, n_3, \ldots$; it produces a batch of multiples $n_1P_1, n_2P_2, n_3P_3, \ldots$. The speed of BBE251 does not rely on any relationships between the inputs; in fact, for fixed-size batches, the software takes constant time, independent of the inputs. BBE251 nevertheless provides speed benefits from handling (n_1, P_1) together with (n_2, P_2) and (n_3, P_3) and so on. BBE251 is faster than the software in [35] for the same field size once the batch size is above 50; for large batches it is more than twice as fast.

Real-world servers bottlenecked by typical elliptic-curve computations can gain speed by collecting the computations into batches, switching to the curve introduced in this paper, and switching to the software introduced in this paper. The batching increases latency by several milliseconds, but in most applications this is not a problem, whereas raw throughput is often critical. BBE251 completes 1048576 scalar multiplications in just 35 seconds using all four cores of a single 2.4GHz Core 2 Quad Q6600 CPU; interference among the cores is negligible. This is not the fastest single-chip scalar-multiplication measurement ever reported in the literature—Güneysu and Paar in [36] reported "more than 37000 point multiplications per second"—but a closer look shows that [36] achieved 37000 224-bit scalar multiplications per second on a \$1000 Xilinx Virtex-4 SX55 containing hundreds of multipliers, while this paper achieves 30000 251-bit scalar multiplications per second on a \$200 Core 2 Quad Q6600.

BBE251 provides several benefits beyond speed. It avoids all data-dependent array indices, all data-dependent branches, etc., and is therefore immune to cache-timing attacks, branch-prediction attacks, etc.; the same security feature was already present in state-of-the-art software for large-characteristic elliptic curves (see, e.g., [12]) but is hard to find for binary curves. BBE251 has been posted (`http://binary.cr.yp.to`) to allow public verification of its accuracy and speed, and has been placed into the public domain to maximize reusability.

How these speeds were achieved: low level. Schoolbook multiplication of two 251-bit polynomials in $\mathbf{F}_2[t]$ takes 125501 bit operations: specifically, $251^2 = 63001$ bit multiplications (ANDs) and $250^2 = 62500$ bit additions (XORs). BBE251 instead uses several layers of Karatsuba and Toom recursions, including some new refinements, reducing the number of bit operations to 33096, approximately 3.79× smaller than 125501. See Section 2 for details.

The conventional wisdom is that bit-operation counts are a poor predictor of software performance, for two critical reasons:

- CPUs handle multiple bits at once. For example, a 32-bit xor is a single instruction, just as fast as a 16-bit xor or an 8-bit xor. It therefore makes no sense to split a 32×32-bit problem into three 16×16-bit problems or to split a 16×16-bit problem into three 8×8-bit problems.

- $\mathbf{F}_2[t]$-multiplication software is bottlenecked by the cost of shifting bits within words, not by the cost of performing arithmetic on bits. For example, extracting two 4-bit pieces from an 8-bit input is not free; it costs two or three instructions, depending on the CPU.

It is widely appreciated that fast multiplication techniques save time in software *for sufficiently large inputs*, but the speedups are generally believed to be rather small at cryptographic sizes—certainly not a $3.79\times$ speedup for 251-bit multiplication. Hankerson, Hernandez, and Menezes in [38, Section 3] say that Karatsuba is "competitive" for 233 bits and for 283 bits but does not actually save time. Bailey and Paar in [10] say that fast polynomial multiplication "yields a 10% speedup in the overall scalar multiplication time" for a particular curve. Brent, Gaudry, Thomé, and Zimmermann in [22] report speedups from several layers of Karatsuba and Toom recursions, but only beyond cryptographic sizes.

In `BBE251`, a 16-bit xor *is* faster than a 32-bit xor, because two 16-bit inputs are packed into the same space as a 32-bit input and handled in parallel. Shift costs are trivially eliminated by the standard technique of "bitslicing": w separate b-bit inputs $i_0 = (i_{0,0}, i_{0,1}, \ldots, i_{0,b-1})$, $i_1 = (i_{1,0}, i_{1,1}, \ldots, i_{1,b-1})$, $\ldots$ are batched, transposed into b separate w-bit vectors $(i_{0,0}, i_{1,0}, \ldots)$, $(i_{0,1}, i_{1,1}, \ldots)$, $\ldots$, $(i_{0,b-1}, i_{1,b-1}, \ldots)$, and then handled without shifts until the end of the computation. These vectors do not fit simultaneously into CPU registers, but `BBE251` arranges operations so that most loads and stores are overlapped with computation.

Bitslicing has been used in cryptography before. Two record-setting examples are Biham's implementation [18] of DES, a standard hardware-friendly bit-oriented cipher that had previously been viewed as very slow in software, and the Matsui–Nakajima implementation [50] of AES, taking just 9.2 cycles per byte on a Core 2. Aoki, Hoshino, and Kobayashi in [9] pointed out that bitsliced field arithmetic saves time for binary elliptic curves—but they were still unable to compete with non-binary elliptic curves. The Pentium III speeds reported in [9, Table 3] for a subfield curve over a field of size 2^{163} are not as fast as the Pentium II speeds reported the same year by Aoki et al. in [8, Table 4] for a curve over a larger non-binary field. The fast multiplication circuits built into current CPUs perform a huge number of bit operations per cycle and are generally perceived as indispensable tools for fast public-key cryptography; it is surprising that bitsliced field arithmetic can set new Diffie–Hellman speed records, outperforming the multiplication circuits.

How these speeds were achieved: high level. There is a limitation to the power of bitslicing, at least in the pure form used in this paper: bitslicing requires all computations to be expressed as straight-line sequences of bit operations. Straight-line computations do not contain data-dependent branches (e.g., "if $P = Q$ then $\ldots$") or data-dependent array indices (e.g., "load $x[i]$, where i is the next bit of the scalar"). Computations that include data-dependent array indices can be simulated by straight-line computations that perform arithmetic on all array elements, and computations that include data-dependent branches can be simulated in an analogous way (recall that the "program counter" is just another array index), but these simulations are slow.

Fortunately, a recent line of research has shown how to carry out some elliptic-curve computations without data-dependent array indices, without data-dependent branches, and without serious loss of speed. These techniques have been advertised as efficiently protecting against various types of software side-channel attacks, such as cache-timing attacks, but the same techniques are also useful for efficient bitslicing. Specifically:

- Single-scalar multiplication, odd characteristic: [12, Theorem 2.1] says that the differential-addition formulas from [52] compute $X_0(nP)$ from $X_0(P)$ on any Montgomery curve having a unique 2-torsion point. Here $X_0(P)$ means x if $P = (x, y)$ and 0 if $P = \infty$. See [13, Section 5] for discussion of more general Montgomery curves.
- Arbitrary group operations, odd characteristic: [14, Theorem 3.3] says that various addition formulas are complete—i.e., have no exceptional cases—for any Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ having non-square d. Complete addition formulas allow complete single-scalar multiplication, complete double-scalar multiplication, etc.
- Arbitrary group operations, characteristic 2: [16, Theorem 4.1] says that various addition formulas in [16] are complete for any binary Edwards curve $d_1(x + y) + d_2(x^2 + y^2) = (x + x^2)(y + y^2)$ having d_2 of trace 1.

The complete differential-addition formulas in [16, Section 7] are reviewed in Section 3 of this paper and are used in `BBE251`. Other approaches, such as mixing bitsliced computation with non-bitsliced computation, do not appear to provide noticeable benefits for the speed of elliptic-curve scalar multiplication and are not discussed in this paper.

The unexpected speed of bitsliced binary-field multiplication can be viewed as motivation to consider bitslicing for a wide variety of higher-level cryptographic computations and for many other problems. This paper's results on binary-elliptic-curve scalar multiplication are one step along this path. The reader is cautioned, however, that bitslicing will provide smaller benefits for computations that rely more heavily on random access to memory.

What about `PCLMULQDQ`? Some CPU designers have begun to realize the potential importance of $\mathbf{F}_2[t]$ for cryptographic applications. Intel announced in April 2008 that its processors will eventually include a "carry-less multiplication" instruction named `PCLMULQDQ`. Intel's white paper [40] says that "carry-less multiplication"—i.e., multiplication in $\mathbf{F}_2[t]$—is particularly important for GCM, a standard for secret-key authenticated encryption relying heavily on multiplications in the binary field $\mathbf{F}_{2^{128}}$; and that "accelerating carry-less multiplication can significantly contribute to achieving high speed secure computing and communication." Gueron and Kounavis estimate in [37] that scalar multiplication on the NIST B-233 elliptic curve would take 220000 cycles using a 9-cycle instruction for "carry-less multiplication," or 70000 cycles using a 3-cycle instruction for "carry-less multiplication." NIST B-233 has a somewhat lower security level than the 251-bit curve used in this paper.

Intel also announced in April 2008 that its processors will, starting in 2010, include 256-bit vector instructions. The initial instruction set will not include many integer instructions but *will* include 256-bit logical operations such as `VXORPS ymm` and `VANDPS ymm`; see [41].

It is clear that `PCLMULQDQ`-based software will easily outperform most software for binary-elliptic-curve computations. However, it is not at all clear that `PCLMULQDQ` will outperform a 256-bit-vector implementation of the techniques introduced in this paper. It is not even clear that `PCLMULQDQ` will outperform the specific `BBE251` software described in this paper!

In any case, one can reasonably speculate that continued improvements in binary-elliptic-curve performance will encourage cryptographic users to shift to binary elliptic curves, increasing the importance of techniques to accelerate computations on those curves. Presumably $\mathbf{Z}$-biased CPUs will continue being sold for years and will continue being used for years after that; the techniques in this paper will clearly remain important for $\mathbf{Z}$-biased CPUs whether or not they are helpful for `PCLMULQDQ` CPUs.

2 Polynomial Multiplication

This section explains how to multiply 251-bit polynomials using 33096 bit operations. More generally, this section presents small explicit upper bounds on $M(n)$, the minimum number of bit operations needed to multiply an n-bit polynomial $f_0 + f_1 t + \cdots + f_{n-1}t^{n-1}$ by another n-bit polynomial $g_0 + g_1 t + \cdots + g_{n-1}t^{n-1}$.

Inputs and outputs are represented in the standard form: $f_0 + f_1 t + \cdots + f_{n-1}t^{n-1}$ is represented as an n-bit string $(f_0, f_1, \ldots, f_{n-1})$; $g_0 + g_1 t + \cdots + g_{n-1}t^{n-1}$ is represented as an n-bit string $(g_0, g_1, \ldots, g_{n-1})$; and the output $h_0 + h_1 t + \cdots + h_{2n-2}t^{2n-2}$ is represented as a $(2n-1)$-bit string $(h_0, h_1, \ldots, h_{2n-2})$.

Comparison to previous work. The definition of polynomial multiplication immediately implies that $M(n) \le \Theta(n^2)$. Karatsuba showed in [45] that $M(n) \le \Theta(n^{\lg 3})$. Toom showed in [64] that $M(n) \le n^{1+o(1)}$, and more precisely $M(n) \le n2^{\Theta(\sqrt{\lg n})}$. Schönhage showed in [61] that $M(n) \le \Theta(n \lg n \lg\lg n)$, by adapting an integer-multiplication method by Schönhage and Strassen in [62]. No better asymptotic bounds are known; Fürer in [32] introduced an asymptotically faster multiplication method for integers, but it is not clear whether the method can be adapted to binary polynomials.

Of course, bounds involving Θ, O, etc. are not explicit. To draw conclusions about $M(251)$, or any other specific $M(n)$, one needs to carefully re-analyze the algorithms used to prove asymptotic bounds. To draw *useful* conclusions one often needs to rethink the algorithm design, looking for constant-factor (and sub-constant-factor) improvements that are not visible in the asymptotics.

Explicit upper bounds do appear in the literature on hardware multipliers. The bound $M(n) \le 2n^2 - 2n + 1$ is easy to find in hardware textbooks. Several cryptographic-hardware papers (see below) have presented explicit upper bounds on $M(n)$ obtained with Karatsuba's method, have pointed out that these upper bounds are considerably below $2n^2 - 2n + 1$ for cryptographically useful sizes of n, and have concluded that hardware multipliers should use Karatsuba's method. XOR and AND do not have identical hardware costs, but analyses weighted by the actual costs come to similar conclusions.

The explicit upper bounds on $M(n)$ obtained in this section are better than anything that can be found in the hardware literature. Here are several examples of the improved bounds:

- $M(128) \leq 11486$. For comparison, Peter and Langendörfer in [57, Table 3] report 14287 bit operations (12100 XORs and 2187 ANDs) for "classic Karatsuba multiplication," 32513 bit operations for schoolbook multiplication, and 13146 bit operations for an "improved iterative Karatsuba."
- $M(163) \leq 16923$. For comparison, Chang, Kim, Park, and Lim in [24] report 21791 bit operations for a "non-redundant Karatsuba-Ofman algorithm."
- $M(193) \leq 21865$. For comparison, Rodríguez-Henríquez and Koç in [59, Section 4.1] report 29725 bit operations (20524 XORs and 9201 ANDs).
- $M(194) \leq 21906$. For comparison, von zur Gathen and Shokrollahi in [67, Section 3] report 26575 bit operations.
- $M(n) \leq (103/18)n^{\lg 3} - 7n + 3/2$ for $n \in \{2, 4, 8, 16, \dots\}$; this bound is not tight. For comparison, Fan, Sun, Gu, and Lam in [29, Table II] report $6.5n^{\lg 3} - 6n + 0.5$ bit operations ($5.5n^{\lg 3} - 6n + 0.5$ XORs and $n^{\lg 3}$ ANDs). Note that $103/18 = 5.722\ldots < 6.5$.
- $M(512) \leq 98018$. For comparison, Rodriguez-Henríquez and Koç in [59, Table 1] report 116191 bit operations (81199 XORs and 34992 ANDs).

As these examples illustrate, switching from this paper's multiplication methods back to the multiplication methods in the hardware literature often increases the number of bit operations by more than 20%. On the other hand, one should not exaggerate the importance of multiplication refinements as a component of this paper's new speed records for binary-elliptic-curve cryptography. Bitsliced binary-Edwards-curve arithmetic, as described in Section 3 of this paper, would still have set new speed records even if this section's multiplication methods had been replaced by the methods in [67].

Schoolbook recursion. One can multiply $f_0 + f_1 t + \cdots + f_n t^n$ by $g_0 + g_1 t + \cdots + g_n t^n$ as follows:

- Recursively multiply $f_0 + f_1 t + \cdots + f_{n-1} t^{n-1}$ by $g_0 + g_1 t + \cdots + g_{n-1} t^{n-1}$.
- Compute $(f_n g_0 + f_0 g_n)t^n + (f_n g_1 + f_1 g_n)t^{n+1} + \cdots + f_n g_n t^{2n}$. This takes $2n + 1$ bit multiplications and n bit additions.
- Add. This takes $n - 1$ bit additions for the coefficients of $t^n, \dots, t^{2n-2}$; the other coefficients do not overlap.

Consequently $M(n+1) \leq M(n) + 4n$. This "schoolbook recursion" bound implies the "schoolbook multiplication" bound $M(n) \leq 2n^2 - 2n + 1$, but schoolbook recursion is—in combination with other recursions—useful for much larger n's than schoolbook multiplication.

Three-way recursion. The well-known **Karatsuba identity**

$$\begin{aligned}(F_0 + F_1 t^n)(G_0 + G_1 t^n) \\ = F_0 G_0 + t^n((F_0 + F_1)(G_0 + G_1) - F_0 G_0 - F_1 G_1) + t^{2n} F_1 G_1\end{aligned}$$

shows how to multiply $2n$-bit polynomials $F_0 + F_1t^n$, $G_0 + G_1t^n$ using three multiplications of n-bit polynomials and a small amount of overhead. What actually appeared as "Theorem 2 (Karatsuba)" in the original Karatsuba–Ofman paper [45] was an integer analogue of the squaring case of the equivalent identity

$$\begin{aligned}&(F_0 + F_1t^n)(G_0 + G_1t^n)\\ &\quad = (1 - t^n)F_0G_0 + t^n(F_0 + F_1)(G_0 + G_1) + (t^{2n} - t^n)F_1G_1.\end{aligned}$$

Either identity easily leads to the bound $M(2n) \le 3M(n) + 8n - 4$ appearing in, e.g., [67, Section 2]. The $8n - 4$ arises as a sum of three components: $2n$ for the additions in $F_0 + F_1$ and $G_0 + G_1$, another $4n - 2$ to subtract F_0G_0 and F_1G_1 from $(F_0 + F_1)(G_0 + G_1)$, and another $2n - 2$ to handle the overlaps between F_0G_0, $t^n \cdots$, and $t^{2n}F_1G_1$.

Much less well known is that the constant 8 here can be improved to 7: specifically, $M(2n) \le 3M(n) + 7n - 3$. What follows is one way to understand the improvement.

A generic quadratic polynomial $H = H_0 + H_1x + H_2x^2$ can be reconstructed from $H(0) = H_0$, $H(1) = H_0 + H_1 + H_2$, and $H(\infty) = H_2$ by the projective Lagrange interpolation formula $H = (1 - x)H(0) + xH(1) + x(x - 1)H(\infty)$. Factoring out $1 - x$ produces the slightly simpler formula

$$H = (1 - x)(H(0) - xH(\infty)) + xH(1).$$

In particular, if H is the product of two generic linear polynomials $F = F_0 + F_1x$ and $G = G_0 + G_1x$, then $H(0) = F(0)G(0) = F_0G_0$, $H(1) = F(1)G(1) = (F_0+F_1)(G_0+G_1)$, and $H(\infty) = F(\infty)G(\infty) = F_1G_1$, so $(F_0+F_1x)(G_0+G_1x) = (1 - x)(F_0G_0 - xF_1G_1) + x(F_0 + F_1)(G_0 + G_1)$. Substitute $x = t^n$ to obtain the **refined Karatsuba identity**

$$(F_0 + F_1t^n)(G_0 + G_1t^n) = (1 - t^n)(F_0G_0 - t^nF_1G_1) + t^n(F_0 + F_1)(G_0 + G_1).$$

For comparison, rewriting the projective Lagrange interpolation formula as $H = H(0) + x(H(1) - H(0) - H(\infty)) + x^2H(\infty)$ leads to the original Karatsuba identity.

Say F_0, G_0 are n-bit polynomials in $\mathbf{F}_2[t]$ and F_1, G_1 are k-bit polynomials in $\mathbf{F}_2[t]$. Here is a cost analysis of the refined Karatsuba identity as a method of computing the product of $F_0 + F_1t^n$ and $G_0 + G_1t^n$:

- Cost $M(n)$: Multiply F_0 by G_0.
- Cost $M(k)$: Multiply F_1 by G_1.
- Cost k, assuming $k \le n$: Add F_0 to F_1.
- Cost k: Add G_0 to G_1.
- Cost $M(n)$: Multiply $F_0 + F_1$ by $G_0 + G_1$.
- Cost $n - 1$, assuming $k \ge n/2$: Subtract $t^nF_1G_1$ from F_0G_0.
- Cost $2k - 1$: Subtract $t^n(F_0G_0 - t^nF_1G_1)$ from $F_0G_0 - t^nF_1G_1$.
- Cost $2n - 1$: Add $t^n(F_0 + F_1)(G_0 + G_1)$.

Consequently $M(n+k) \le 2M(n) + M(k) + 4k + 3n - 3$ if $n/2 \le k \le n$.

Notice that this computation does not follow the traditional structure of first computing the coefficients H_0, H_1, H_2 and then computing $H(t^n) = H_0 + H_1 t^n + H_2 t^{2n}$. In particular, the middle coefficient $H_1 = (F_0 + F_1)(G_0 + G_1) - F_0 G_0 - F_1 G_1$ is not an intermediate result in this computation.

I posted the $M(2n) \le 3M(n) + 7n - 3$ bound (and the resulting $M(n) \le (103/18)n^{\lg 3} - 7n + 3/2$ bound for $n \in \{2, 4, 8, \ldots\}$) in [11, page 7] in 2000, but I am formally announcing the idea here for the first time. The simplified-Lagrange-interpolation explanation has not appeared anywhere, even informally, and is reused below for improvements in five-way recursion.

Five-way recursion. A generic quartic polynomial $H = H_0 + H_1 x + H_2 x^2 + H_3 x^3 + H_4 x^4$ over $\mathbf{F}_2$ can be reconstructed from the values $H(0)$, $H(1)$, $H(t)$, $H(t+1)$, $H(\infty)$ by the projective Lagrange interpolation formula

$$\begin{aligned} H = {}& H(0)\frac{(x+1)(x+t)(x+t+1)}{t(t+1)} + H(1)\frac{x(x+t)(x+t+1)}{(1+t)t} \\ &+ H(t)\frac{x(x+1)(x+t+1)}{t(t+1)} + H(t+1)\frac{x(x+1)(x+t)}{(t+1)t} \\ &+ H(\infty)x(x+1)(x+t)(x+t+1). \end{aligned}$$

Easy manual simplification produces the (perhaps not optimal) formula

$$H = U + H(\infty)(x^4 + x) + \frac{(U + V + H(\infty)(t^4 + t))(x^2 + x)}{t^2 + t}$$

where $U = H(0) + (H(0) + H(1))x$ and $V = H(t) + (H(t) + H(t+1))(x + t)$. This formula, in turn, leads to the following new algorithm to compute the product of $F_0 + F_1 t^n + F_2 t^{2n}$ and $G_0 + G_1 t^n + G_2 t^{2n}$, where F_0, F_1, G_0, G_1 are n-bit polynomials and F_2, G_2 are k-bit polynomials:

- Cost $M(n)$: Compute $H(0) = F_0 G_0$.
- Cost $M(k)$: Compute $H(\infty) = F_2 G_2$.
- Cost $n + k$, assuming $k \le n$: Compute $F_0 + F_1 + F_2$.
- Cost $n + k$: Compute $G_0 + G_1 + G_2$.
- Cost $M(n)$: Compute $H(1) = (F_0 + F_1 + F_2)(G_0 + G_1 + G_2)$.
- Cost $n - 1$, or k if $k \le n - 1$: Compute $F_1 t + F_2 t^2$.
- Cost $n - 1$, or k if $k \le n - 1$: Compute $G_1 t + G_2 t^2$.
- Cost $n - 1$: Compute $F_0 + (F_1 t + F_2 t^2)$.
- Cost $n - 1$: Compute $G_0 + (G_1 t + G_2 t^2)$.
- Cost $M(n+2)$, or $M(n+1)$ if $k \le n - 1$: Compute $H(t)$, the product of $F_0 + (F_1 t + F_2 t^2)$ and $G_0 + (G_1 t + G_2 t^2)$.
- Cost $n - 1$: Compute $(F_0 + F_1 + F_2) + (F_1 t + F_2 t^2)$.
- Cost $n - 1$: Compute $(G_0 + G_1 + G_2) + (G_1 t + G_2 t^2)$.
- Cost $M(n+2)$, or $M(n+1)$ if $k \le n - 1$: Compute $H(t+1)$, the product of $(F_0 + F_1 + F_2) + (F_1 t + F_2 t^2)$ and $(G_0 + G_1 + G_2) + (G_1 t + G_2 t^2)$.

- Cost $2n+1$: Compute $H(t)+H(t+1)$. The coefficients of t^{2n+2} and t^{2n+1} in $H(t+1)$ are the same as the coefficients of t^{2n+2} and t^{2n+1} in $H(t)$, so this sum has degree at most $2n$. (For the same reason, some work could have been saved in the computation of $H(t+1)$.)
- Cost $3n+4$, or $3n+2$ if $k \le n-1$: Compute $V = H(t) + (H(t)+H(t+1))(t^n+t)$. Note that $\deg V \le 3n$.
- Cost $3n-2$: Compute $U = H(0)+(H(0)+H(1))t^n$. Note that $\deg U \le 3n-2$.
- Cost $4k+3n-3$, assuming $n \ge 2$: Compute $W = U + V + H(\infty)(t^4+t)$. Note that $\deg W \le 3n$.
- Cost $3n-2$: Compute $W/(t^2+t)$. This division is exact: $W/(t^2+t) \in \mathbf{F}_2[t]$. (For the same reason, some work could have been skipped in the computation of W.)
- Cost $5n+2k-4$: Compute $H(\infty)(t^{4n}+t^n) + (W/(t^2+t))(t^{2n}+t^n) + U$.

Consequently $M(3n) \le 3M(n)+2M(n+2)+35n-12$ if $n \ge 2$, and $M(2n+k) \le 2M(n)+M(k)+2M(n+1)+25n+10k-12$ if $1 \le k \le n-1$.

The previous state of the art, building on ideas by Zimmermann and Quercia, Weimerskirch and Paar [68], and Montgomery [53], was an algorithm by Bodrato in [19] to compute H_0, H_1, H_2, H_3, H_4 given $H(0), H(1), H(t), H(t+1), H(\infty)$ using 9 additions, one multiplication by t^3+1, one division by t, one division by $t+1$, and one division by t^2+t. The total overhead was about $38n$ operations: $10n$ to compute $F(0), F(1), F(t), F(t+1), F(\infty)$ and $G(0), G(1), G(t), G(t+1), G(\infty)$; $24n$ for Bodrato's computation of H_0, H_1, H_2, H_3, H_4; and $4n$ to reconstruct H from H_0, H_1, H_2, H_3, H_4. The separate coefficients H_0, H_1, H_2, H_3, H_4 turn out to be a distraction, as in the Karatsuba case; this section does better by constructing H directly, exploiting the polynomial structure visible in the projective Lagrange interpolation formula.

Two-level seven-way recursion. Consider the problem of multiplying two degree-3 polynomials. Apply the refined Karatsuba identity three times, factor out $1-x$, and substitute $x = t^n$, to obtain the identity

$$\begin{aligned}(F_0 + F_1t^n + F_2t^{2n} + F_3t^{3n})&(G_0 + G_1t^n + G_2t^{2n} + G_3t^{3n})\\ = (1-t^{2n})&((1-t^n)(F_0G_0 - t^nF_1G_1 - t^{2n}F_2G_2 + t^{3n}F_3G_3)\\ &\qquad + t^n(F_0+F_1)(G_0+G_1) - t^{3n}(F_2+F_3)(G_2+G_3))\\ &+ t^{2n}(F_0+F_2+(F_1+F_3)t^n)(G_0+G_2+(G_1+G_3)t^n).\end{aligned}$$

Cost evaluation for polynomials with $3n+k$ coefficients, assuming $k \ge n/2$:

- Cost $M(n)$: Multiply F_0 by G_0.
- Cost $M(n)$: Multiply F_1 by G_1.
- Cost $M(n)$: Multiply F_2 by G_2.
- Cost $M(k)$: Multiply F_3 by G_3.
- Cost $3n-3$: Compute $U = F_0G_0 - t^nF_1G_1 - t^{2n}F_2G_2 + t^{3n}F_3G_3$.
- Cost $2n+2k-1$: Compute $(1-t^n)U$.
- Cost $2n+M(n)$: Multiply F_0+F_1 by G_0+G_1.
- Cost $2k+M(n)$: Multiply F_2+F_3 by G_2+G_3.

- Cost $4n-2$: Compute $V = (1-t^n)U + t^n(F_0+F_1)(G_0+G_1) - t^{3n}(F_2+F_3)(G_2+G_3)$.
- Cost $2n+2k+M(2n)$: Multiply $F_0+F_2+(F_1+F_3)t^n$ by $G_0+G_2+(G_1+G_3)t^n$.
- Cost $6n+2k-2$: Compute $(1-t^{2n})V + t^{2n}(F_0+F_2+(F_1+F_3)t^n)(G_0+G_2+(G_1+G_3)t^n)$.

Hence $M(3n+k) \le M(2n)+5M(n)+M(k)+19n+8k-8$ if $n/2 \le k \le n$. For example, $M(4n) \le M(2n)+6M(n)+27n-8$. This is $n-1$ smaller than what would have been obtained by straightforwardly applying the refined Karatsuba identity without factoring out $1-x$.

Optimization. One can build a table of upper bounds on $M(1),\dots,M(n)$ by recursively building a table of upper bounds on $M(1),\dots,M(n-1)$ and then mechanically checking what the inequalities in this section say about $M(n)$. This computation reaches $M(251)$ in negligible time. One can slightly improve many of the upper bounds by mechanically removing redundant computations (such as the equal top coefficients of $H(t)$ and $H(t+1)$ in five-way recursion) from straight-line multiplication code.

My web page `http://binary.cr.yp.to/m.html` presents a table of upper bounds on $M(1),\dots,M(1000)$ obtained in this way. Each upper bound is accompanied by straight-line multiplication code that has been computer-verified to multiply correctly and to use exactly the specified number of bit operations.

Often an input to one multiplication is reused in a subsequent multiplication; for example, w_1 in Section 3 participates in many multiplications. One can save time by caching evaluations of that input, such as F_0+F_1 above. To properly optimize this reuse one should define, e.g., $M_2(n)$ as the cost of multiplying a single n-bit input by two n-bit inputs (serially), and then optimize $M_2(n)$ analogously to $M(n)$.

The reader is cautioned that there are many more multiplication methods in the literature: for example, more Toom variants, FFTs, etc. Analyzing, refining, and combining these methods would improve the bounds on $M(n)$ for many integers n, perhaps including $n=251$. Most of the relevant methods are surveyed in [22] but have not yet been optimized for bit operations.

3 Elliptic-Curve Scalar Multiplication

This section reviews binary Edwards curves; discusses this paper's selection of a particular binary Edwards curve; and analyzes the speed of computation of scalar multiples on that curve.

Review of binary Edwards curves. A **binary Edwards curve** over a binary finite field k is a curve of the form $d_1(x+y)+d_2(x^2+y^2) = (x+x^2)(y+y^2)$ where $d_1 \in k-\{0\}$ and $d_2 \in k-\{d_1^2+d_1\}$. This curve shape was introduced by Bernstein, Lange, and Rezaeian Farashahi in [16] as a characteristic-2 analogue to the curve shape introduced by Edwards in [28].

A binary Edwards curve is **complete** if d_2 has trace 1, i.e., if d_2 cannot be written as c^2+c for any $c \in k$. The paper [16] proves that every ordinary elliptic curve over k is birationally equivalent over k to a complete binary Edwards curve if $\#k \geq 8$, and that various addition formulas on the complete binary Edwards curve have no exceptional cases.

The case $d_1 = d_2$ allows various speedups presented in [16]. For example, it allows differential addition and doubling—a single step in a "Montgomery ladder"—using four squarings in k, two multiplications by d_1, and five more multiplications in k. The general case would need four squarings in k, four multiplications by parameters, and six more multiplications in k.

The selected curve. Define $k = \mathbf{F}_{2^{251}} = \mathbf{F}_2[t]/(t^{251} + t^7 + t^4 + t^2 + 1)$. Define $d \in k$ as $t^{57} + t^{54} + t^{44} + 1$; note that d has trace 1. Define E as the binary Edwards curve $d(x + x^2 + y + y^2) = (x + x^2)(y + y^2)$.

This curve is birationally equivalent to the Weierstrass curve $v^2 + uv = u^3 + (d^2 + d)u^2 + d^8$ by $(x, y) \mapsto (u, v) = (d^3(x+y)/(xy+d(x+y)), d^3(x/(xy+d(x+y)) + d + 1))$. See [16, Section 2].

The main task considered here is scalar multiplication $n, P \mapsto nP$ in the group $E(k) = \{(x, y) \in k^2 : d(x + x^2 + y + y^2) = (x + x^2)(y + y^2)\}$, with neutral element $(0, 0)$. Note that this group does not have any points at infinity. See [16] for further discussion of the group law.

Security issues in curve selection. This curve satisfies all of the standard criteria for high-security curves:

- The curve has near-prime order. Specifically, the curve has order $4p_1$ where p_1 is the prime $2^{249} + 17672450755679567125975931502191870417$.
- The twist of the curve has near-prime order, specifically order $2p_2$ where p_2 is the prime $2^{250} - 35344901511359134251951863004383740833$.
- The primes are large enough for high security: generic discrete-logarithm algorithms use approximately 2^{124} group operations on average.
- Avoiding subfields: The j-invariant $1/d^8$ generates the field k.
- Avoiding small discriminants: $(2^{251}+1-4p_1)^2-2^{253}$ is divisible by the large prime $((2^{251} + 1 - 4p_1)^2 - 2^{253})/(-83531196553759)$ exactly once.
- Avoiding pairing attacks: The multiplicative order of 2^{251} modulo p_1 is not small: in fact, it is $(p_1 - 1)/2$. The multiplicative order of 2^{251} modulo p_2 is not small: in fact, it is $(p_2 - 1)/2$.
- Avoiding the GHS attack: The extension degree 251 is a prime, so the only nontrivial subfield of k is $\mathbf{F}_2$. GHS genera over $\mathbf{F}_2$ cannot be small: in fact, they are at least 2^{49}, since the multiplicative order of 2 modulo 251 is 50. See [51].

The curve used in this paper was found by a search through various possibilities for d. Most choices of d fail the near-prime-order requirement, and most of the remaining choices of d fail the near-prime-twist-order requirement, but there are still many suitable possibilities. An easy computation with the Magma computer-algebra system [20] located a few suitable trinomials, many suitable quadrinomials, etc. The first trinomial found was $t^{141} + t^{28} + 1$, the first quadrinomial found was $t^{57} + t^{54} + t^{44} + 1$ (although $(t^{222} + 1)(t^{21} + 1)$ is an interesting alternative), and the first pentanomial found was $t^{23} + t^{16} + t^{15} + t + 1$.

Some standards omit, or weaken, some of the criteria listed above, for several reasons:

- Some of the criteria do not have known benefits. For example, subfield curves produce only a small loss of security, which can be corrected by a small increase in field size. The small-discriminant criterion is even more difficult to defend; there is no known attack that exploits small discriminants, and there are reasons to guess that randomly chosen large-discriminant curves are *more dangerous* than small-discriminant curves. See [47, Sections 11.1–11.3].
- Sometimes the criteria have disadvantages. For example, some criteria have to be weakened by anyone who wants to allow curves with special algebraic structures, such as "pairing-friendly curves," "Koblitz curves," "Gallant–Lambert–Vanstone curves," and the new "Galbraith–Lin–Scott curves." See, e.g., [34], [33], and [39].
- One of the criteria, twist security, has protocol-level benefits that were not visible in the traditional study of the elliptic-curve discrete-logarithm problem. Twist security has, as a result, often been neglected even in situations where it has no disadvantages. For further discussion of twist security see [44, Section 4], [21, Section 4], [25, Section 4.1], [12, Section 3], and [31, Section 5].

This paper's selection of a curve meeting all the security criteria should not be interpreted as criticism of curves that meet fewer security criteria. One should expect some of those curves, when combined with the techniques in this paper, to achieve even better speeds than the speeds reported in this paper.

Speed issues in curve selection. Even within the restricted pool of curves meeting all of the security criteria discussed above, there are still considerable variations in speed. Standard practice is to focus on the highest-speed curves.

In particular, the fastest elliptic-curve-scalar-multiplication methods involve many multiplications by curve coefficients; it is standard practice to choose these coefficients to be "small." The exact definition of "small" varies but is aimed at speeding up multiplications by these coefficients. For example:

- Weierstrass curves: IEEE Standard P1363 chooses curves $y^2 = x^3 - 3x + b$ to "provide the fastest arithmetic on elliptic curves"; see [2, Section A.9]. Chudnovsky and Chudnovsky had pointed out in [26] that choosing a small coefficient a in $y^2 = x^3 + ax + b$ saves time in elliptic-curve scalar multiplication, and that the particular choice $a = -3$ saves even more time. NIST's standard curves were chosen by the recipe specified in IEEE Standard P1363; see [1, Appendix 6, Section 1.4].
- Montgomery curves: The curve "Curve25519" specified in [12] is the curve $y^2 = x^3 + 486662x^2 + x$ modulo $2^{255} - 19$. Montgomery had pointed out in [52] that his fast differential-addition formulas for $y^2 = x^3 + ax^2 + x$ involve multiplications by $(a + 2)/4$ and benefit from $(a + 2)/4$ being small.
- Binary Edwards curves: [16] makes the analogous suggestion to choose a small parameter d for the curve $d(x + x^2 + y + y^2) = (x + x^2)(y + y^2)$.

This paper chooses $d = t^{57} + t^{54} + t^{44} + 1$, combining a small degree with a small number of terms. Multiplication of a 251-bit polynomial by the quadrinomial $t^{57} + t^{54} + t^{44} + 1$ in $\mathbf{F}_2[t]$ uses only $3 \cdot 251 - 57 = 696$ bit operations, and reduction of the 308-bit product modulo $t^{251} + t^7 + t^4 + t^2 + 1$ uses only 200 bit operations (slightly better than the obvious bound $4 \cdot 57 = 228$ since $t^4, t^2, 1$ are evenly spaced), for a total of just 896 bit operations to multiply by d.

Differential addition and doubling on binary Edwards curves. The following formulas are the "affine $d_1 = d_2$" and "mixed $d_1 = d_2$" formulas from [16, Section 7], repeated here to keep this paper self-contained.

For each point $P = (x, y) \in E(k)$ define $w(P) = x + y$. Then $w(2P) = 1 + d/(d + w(P)^2 + w(P)^4)$, and more generally $w(Q + P) + w(Q - P) = 1 + d/(d + w(P)w(Q)(1 + w(P))(1 + w(Q)))$. The denominators here are never zero.

The following formulas use four squarings, two multiplications by d, and five more multiplications to compute $w(2P), w(Q + P)$ as fractions $W_4/Z_4, W_5/Z_5$, given $w(P), w(Q)$ as fractions $W_2/Z_2, W_3/Z_3$ and given $w(Q - P)$ as an element $w_1 \in k$:

$$C = W_2 \cdot (Z_2 + W_2); \qquad W_4 = C^2; \qquad Z_4 = d(Z_2^2)^2 + W_4;$$
$$V = C \cdot W_3 \cdot (Z_3 + W_3); \quad Z_5 = V + d(Z_2 \cdot Z_3)^2; \quad W_5 = V + Z_5 \cdot w_1.$$

This operation is called **mixed differential addition and doubling**.

Conditional swaps. This paper uses a scalar-multiplication strategy introduced by Montgomery in [52, Section 10.3.1], often called the "Montgomery ladder." The most important step is **conditionally swapped mixed differential addition and doubling**. This means computation of $w(2P), w(Q + P)$ as fractions $W_4/Z_4, W_5/Z_5$ if $\beta = 0$, and computation of $w(P + Q), w(2Q)$ as fractions $W_4/Z_4, W_5/Z_5$ if $\beta = 1$. The inputs are $w(P), w(Q)$ as fractions $W_2/Z_2, W_3/Z_3$ as above; $w(Q - P)$ as an element $w_1 \in k$ as above; and an extra bit $\beta \in \{0, 1\}$.

The standard way to handle $\beta = 1$ is to first swap $W_2/Z_2, W_3/Z_3$, then proceed with the original computation, and finally swap $W_4/Z_4, W_5/Z_5$. The first swap exactly reverses the roles of P and Q, since $w(P - Q) = w(Q - P)$; the original computation therefore produces $w(2Q), w(P + Q)$; and the final swap produces $w(P + Q), w(2Q)$ as desired.

The standard way to swap W_2 and W_3 conditionally on β without branching is to replace (W_2, W_3) by $(W_2 + \beta(W_3 - W_2), W_3 - \beta(W_3 - W_2))$. Similar comments apply to (Z_2, Z_3), (W_4, W_5), and (Z_4, Z_5).

Scalar multiplication. The last step in scalar multiplication computes $w(nP)$, $w(nP + P)$ as fractions, starting from $w(\lfloor n/2 \rfloor P), w(\lfloor n/2 \rfloor P + P)$ as fractions and $w(P)$ as an element of k. This is an example of conditionally swapped mixed differential addition and doubling, where β is the bottom bit of n.

The previous step produces $w(\lfloor n/2 \rfloor P), w(\lfloor n/2 \rfloor P + P)$ as fractions starting from $w(\lfloor n/4 \rfloor P), w(\lfloor n/4 \rfloor P + P)$ as fractions and $w(P)$ as an element of k. This is the same computation, except that n is replaced by $\lfloor n/2 \rfloor$; i.e., β is the second bit of n. The conditional swap at the end of this step is followed immediately by, and can be profitably merged with, the conditional swap at the beginning of the next step.

Similar comments apply to earlier steps. If the target scalar n is known to be between 0 and $2^b - 1$ then one can use a sequence of b steps. The first step produces $w(\lfloor n/2^{b-1} \rfloor P), w(\lfloor n/2^{b-1} \rfloor P + P)$ from $w(\lfloor n/2^b \rfloor P), w(\lfloor n/2^b \rfloor P + P)$; i.e., from $0, w(P)$.

To summarize: Given $w(P) \in k$ and a b-bit scalar n, this scalar-multiplication method uses b conditionally swapped mixed differential additions and doublings to produce (W, Z) such that $W/Z = w(nP)$. The $2b$ conditional swaps can be merged into just $b + 1$ conditional swaps.

Postprocessing. One can divide W by Z, obtaining $w(nP) \in k$, by computing $WZ^{2^{251}-2}$ with (e.g.) 250 squarings and 11 more multiplications. The multiplications produce Z^3, Z^7, Z^{2^6-1}, $Z^{2^{12}-1}$, $Z^{2^{24}-1}$, $Z^{2^{25}-1}$, $Z^{2^{50}-1}$, $Z^{2^{100}-1}$, $Z^{2^{125}-1}$, $Z^{2^{250}-1}$, and $WZ^{2^{251}-2}$.

A small extra computation shown in [16, Section 7], using the x and y coordinates of P separately, would produce the x and y coordinates of nP separately; but the w coordinate is adequate for elliptic-curve Diffie–Hellman. One can also check directly that w corresponds to a curve point by checking that $d/(w + w^2)$ has trace 0 and that $w + w^2$ times the half-trace of $d/(w + w^2)$ has trace 0. These computations are much faster than scalar multiplication and are not discussed further in this paper.

Instead of inverting Z at the end of the computation one can use affine coordinates, eliminating some multiplications at the cost of an inversion in each step. Inversions are well known to benefit from batching, thanks to "Montgomery's trick" from [52, page 260]. However, each inversion still costs slightly more than 3 multiplications, wiping out most if not all of the gain. Montgomery in [52, page 261] compared batched affine coordinates to projective coordinates in the non-binary case and reported negligible performance differences. One should not expect affine coordinates to provide large savings in the binary case.

Performance: bit operations. A 251-bit single-scalar multiplication as described here involves 44679665 bit operations; this number has been computer-verified. The main cost is 43011084 bit operations for 1266 field multiplications (1255 in the main loop and 11 in the final division). Each multiplication uses 33974 bit operations: 33096 bit operations for 251-bit multiplication in $\mathbf{F}_2[t]$, and 878 bit operations to reduce the 501-bit product modulo $t^{251} + t^7 + t^4 + t^2 + 1$. The other costs are as follows:

- 397518 bit operations for 1254 squarings (1004 in the main loop and 250 in the final division), each using 317 bit operations;
- 449792 bit operations for 502 multiplications by d, each using 896 bit operations;
- 315005 bit operations for 1255 additions, each using 251 bit operations; and
- 506266 bit operations for 1004 conditional swaps, which at the cost of 250 bit operations are merged into 504 conditional swaps, each costing 1004 bit operations.

In some protocols the 251-bit scalar is always a multiple of 4, allowing slight speedups. In other protocols a 249-bit scalar is adequate, allowing slight further speedups.

Performance: cycles. The `BBE251` software reads a batch of scalars $n_1, \ldots, n_{128}$ and a batch of curve points $P_1, \ldots, P_{128}$ and computes a batch of multiples $n_1 P_1, \ldots, n_{128} P_{128}$. Each scalar is represented as a 32-byte string in little-endian form. Each curve point is represented as a field element w as described in the previous section; w is, in turn, represented as a 32-byte string.

One might think that writing fast software for this computation on a Core 2 CPU is a simple matter of generating code for the bit operations described in this paper, replacing XORs and ANDs with a C compiler's intrinsic `_mm_xor_si128` and `_mm_and_si128` operations on 128-bit vectors. A single core of the CPU can carry out three of the corresponding `PXOR` and `PAND` instructions per cycle, so 44 million bit operations should be completed in about 15 million cycles—under 120000 cycles per input. Transposing the n's and P's into bitsliced form, and transposing the results out of bitsliced form, takes negligible time.

There is, however, a critical bottleneck in any straightforward implementation: namely, load throughput. In one cycle the CPU can carry out three operations on six vectors *in registers*; but loading those six vectors from memory into registers costs six cycles—the Core 2 performs only one load in each cycle. The results of the three operations are ready to be used for further operations in the next cycle, so one can imagine loading (e.g.) 56 input vectors for a 28-bit multiplication, carrying out all 956 bit operations for the multiplication, and then storing the final outputs; but the Core 2 has only 16 128-bit vector registers.

Recursive multiplication methods such as Karatsuba's method might seem to be ideal for reducing loads and stores, since they split larger multiplication problems into smaller multiplication problems that fit into registers. However, the first step in decomposing a $2n$-bit multiplication into n-bit multiplications is to add $2n$ vectors to another $2n$ vectors—and the $2n/3$ cycles for these additions are swamped by $4n$ cycles for loads. Similar comments apply to the recombination of $(2n-1)$-bit products into a $(4n-1)$-bit product.

Further contributing to the memory pressure is the fact that the Core 2's vector instructions are two-operand instructions such as "replace a with $a+b$," not three-operand instructions such as "replace c with $a+b$." Copying a to c, in situations where a and b need to be reused, takes away one of the three `XOR/AND` slots available in a cycle. Copying a to c via memory uses an extra load.

`BBE251` takes several measures to reduce the number of loads and stores. Most importantly, it merges decompositions and recombinations across multiple layers of recursion, reusing sums while they are still in registers. As a simple example, adding $(a_0, \ldots, a_{2n-1})$ to $(a_{2n}, \ldots, a_{4n-1})$ takes $4n$ loads and $2n$ additions; subsequently adding $(a_0, \ldots, a_{n-1})$ to $(a_n, \ldots, a_{2n-1})$, adding $(a_{2n}, \ldots, a_{3n-1})$ to $(a_{3n}, \ldots, a_{4n-1})$, and adding $(a_0+a_{2n}, \ldots, a_{n-1}+a_{3n-1})$ to $(a_n+a_{3n}, \ldots, a_{2n-1}+a_{4n-1})$ would take $6n$ loads and $3n$ additions; but performing all of these operations together reduces the $10n$ loads to $4n$ loads and $2n$ copies.

The current version of BBE251 merges most operations across two levels of recursion, and takes fewer than 44 million cycles, although still many more than the target of 15 million. It is not yet clear how close the correlations are between optimized bit-operation counts and optimized cycle counts, but it is clear that schoolbook multiplication could not have been competitive with BBE251. Larger-scale load/store elimination is underway and can be expected to further improve BBE251's performance.

References

1. Digital signature standard (DSS). Federal Information Processing Standard 186-2. National Institute of Standards and Technology (2000), `http://csrc.nist.gov/publications/fips/`, Citations in this document: § 3
2. Standard specifications for public key cryptography. IEEE, Los Alamitos (2000); Citations in this document: §3
3. Information theory workshop, ITW 2006, Chengdu. IEEE, Los Alamitos (2006), See [67]
4. SPEED: software performance enhancement for encryption and decryption (2007), `http://www.hyperelliptic.org/SPEED`, See [35]
5. Design, automation & test in Europe conference & exhibition, 2007. In: DATE 2007. IEEE, Los Alamitos (2007), See [57]
6. Fifth international conference on information technology: new generations (ITNG 2008), Las Vegas, Nevada, USA, April 7-8, 2008. IEEE, Los Alamitos (2008), See [37]
7. Fifth workshop on fault diagnosis and tolerance in cryptography (FDTC 2008). IEEE, Los Alamitos (2008), See [31]
8. Aoki, K., Hoshino, F., Kobayashi, T.: A cyclic window algorithm for ECC defined over extension fields. In: [58], pp. 62–73 (2001); Citations in this document: §1
9. Aoki, K., Hoshino, F., Kobayashi, T., Oguro, H.: Elliptic curve arithmetic using SIMD. In: [27], pp. 235–247 (2001), Citations in this document: §1, §1
10. Bailey, D.V., Paar, C.: Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. Journal of Cryptology 14, 153–176 (2001); ISSN 0933-2790, Citations in this document: §1
11. Bernstein, D.J.: Fast multiplication (2000), `http://cr.yp.to/talks.html#2000.08.14`, Citations in this document: §2
12. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: [69], pp. 207–228 (2006), `http://cr.yp.to/papers.html#curve25519`, Citations in this document: §1, §1, §1, §3, §3
13. Bernstein, D.J.: Can we avoid tests for zero in fast elliptic-curve arithmetic (2006), `http://cr.yp.to/papers.html#curvezero`, Citations in this document: §1
14. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: [49], pp. 29–50 (2007), `http://cr.yp.to/papers.html#newelliptic`, Citations in this document: §1
15. Bernstein, D.J., Lange, T. (eds.): eBACS: ECRYPT Benchmarking of Cryptographic Systems (2009), `http://bench.cr.yp.to` (accessed June 3, 2009); Citations in this document: §1
16. Bernstein, D.J., Lange, T., Farashahi, R. R.: Binary Edwards curves. In: [55], pp. 244–265 (2008), `http://cr.yp.to/papers.html#edwards2`, Citations in this document: §1, §1, §1, §3, §3, §3, §3, §3, §3, §3, §3

17. Biham, E. (ed.): FSE 1997. LNCS, vol. 1267. Springer, Heidelberg (1997); ISBN 3-540-63247-6, See [18]
18. Biham, E.: A fast new DES implementation in software. In: [17], pp. 260–272 (1997); Citations in this document: §1
19. Bodrato, M.: Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In: [23], pp. 116–133 (2007), `http://bodrato.it/papers/#WAIFI2007`, Citations in this document: §2
20. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. Journal of Symbolic Computation 24, 235–265 (1997); Citations in this document: §3
21. Boyd, C., Montague, P., Nguyen, K.: Elliptic curve based password authenticated key exchange protocols. In: [66], pp. 487–501 (2001), `http://sky.fit.qut.edu.au/~boydc/papers/`, Citations in this document: §3
22. Brent, R.P., Gaudry, P., Thomé, E., Zimmermann, P.: Faster multiplication in GF(2)[x]. In: [65], pp. 153–166, `http://wwwmaths.anu.edu.au/~brent/pub/pub232.html`, Citations in this document: §1, §2
23. Carlet, C., Sunar, B. (eds.): WAIFI 2007. LNCS, vol. 4547. Springer, Heidelberg (2007); ISBN 978-3-540-73073-6, See [19]
24. Chang, N.S., Kim, C.H., Park, Y.-H., Lim, J.: A non-redundant and efficient architecture for Karatsuba-Ofman algorithm. In: [70], pp. 288–299 (2005); Citations in this document: §2
25. Chevassut, O., Fouque, P.-A., Gaudry, P., Pointcheval, D.: The Twist-AUgmented technique for key exchange. In: [69], pp. 410–426 (2006), `http://www.loria.fr/~gaudry/papers.en.html`, Citations in this document: §3
26. Chudnovsky, D.V., Chudnovsky, G.V.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Advances in Applied Mathematics 7, 385–434 (1986); MR 88h:11094, Citations in this document: §3
27. Davida, G.I., Frankel, Y. (eds.): ISC 2001. LNCS, vol. 2200. Springer, Heidelberg (2001); ISBN 978-3-540-42662-2, See [9]
28. Edwards, H.M.: A normal form for elliptic curves. Bulletin of the American Mathematical Society 44, 393–422 (2007), `http://www.ams.org/bull/2007-44-03/S0273-0979-07-01153-6/home.html`, Citations in this document: §3
29. Fan, H., Sun, J., Gu, M., Lam., K.-Y.: Overlap-free Karatsuba-Ofman polynomial multiplication algorithms for hardware implementations (October 7, 2008), `http://eprint.iacr.org/2007/393`, Citations in this document: §2
30. Fong, K., Hankerson, D., López, J., Menezes, A.: Field inversion and point halving revisited. IEEE Transactions on Computers 53, 1047–1059 (2004), `http://www.cacr.math.uwaterloo.ca/techreports/2003/tech_reports2003.html`, ISSN 0018–9340, Citations in this document: §1
31. Fouque, P.-A., Lercier, R., Réal, D., Valette, F.: Fault attack on elliptic curve with Montgomery ladder implementation. In: [7], pp. 92–98 (2008), `http://www.di.ens.fr/~fouque/index-pub.html`, Citations in this document: §3
32. Fürer, M.: Faster integer multiplication. In: [42], pp. 57–66 (2007), `http://www.cse.psu.edu/~furer/`, Citations in this document: §2
33. Galbraith, S., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. In: [43], pp. 518–535 (2009), `http://eprint.iacr.org/2008/194`, Citations in this document: §1, §3
34. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: [46], pp. 190–200 (2001), MR 2003h:14043, Citations in this document: §3

35. Gaudry, P., Thomé, E.: The mpFq library and implementing curve-based key exchanges. In: [4], pp. 49–64 (2007), http://www.loria.fr/~gaudry/papers.en.html, Citations in this document: §1, §1, §1
36. Güneysu, T., Paar, C.: Ultra high performance ECC over NIST primes on commercial FPGAs. In: [55], pp. 62–78 (2008); Citations in this document: §1, §1
37. Gueron, S., Kounavis, M.E.: A technique for accelerating characteristic 2 elliptic curve cryptography. In: [6], pp. 265–272 (2008); Citations in this document: §1
38. Hankerson, D., Hernandez, J.L., Menezes, A.: Software implementation of elliptic curve cryptography over binary fields. In: [48], pp. 1–24 (2000), http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-42.ps, Citations in this document: §1, §1
39. Hankerson, D., Karabina, K., Menezes., A.: Analyzing the Galbraith–Lin–Scott point multiplication method for elliptic curves over binary fields (2008), http://eprint.iacr.org/2008/334, Citations in this document: §1, §3
40. Intel Corporation, Carry-less multiplication and its usage for computing the GCM mode (2008), http://software.intel.com/en-us/articles/carry-less-multiplication-and-its-usage-for-computing-the-gcm-mode, Citations in this document: §1
41. Intel Corporation, Intel Advanced Vector Extensions programming reference (2008), http://softwarecommunity.intel.com/isn/downloads/intelavx/Intel-AVX-Programming-Reference-31943302.pdf, Citations in this document: §1
42. Johnson, D.S., Feige, U. (eds.): Proceedings of the 39th annual ACM symposium on theory of computing, San Diego, California, USA, June 11–13. Association for Computing Machinery, New York (2007); ISBN 978–1–59593–631–8, See [32]
43. Joux, A. (ed.): EUROCRYPT 2009. LNCS, vol. 5479. Springer, Heidelberg (2009); ISBN 978-3-642-01000-2, See [33]
44. Kaliski Jr., B.S.: One-way permutations on elliptic curves. Journal of Cryptology 3, 187–199 (1991), Citations in this document: §3
45. Karatsuba, A.A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics Doklady 7, 595–596 (1963), http://cr.yp.to/bib/entries.html#1963/karatsuba, ISSN 0038–5689, Citations in this document: §2, §2
46. Kilian, J. (ed.): CRYPTO 2001. LNCS, vol. 2139. Springer, Heidelberg (2001); ISBN 3-540-42456-3. MR 2003d:94002, See [34]
47. Koblitz, A.H., Koblitz, N., Menezes, A.: Elliptic curve cryptography: the serpentine course of a paradigm shift (2008), http://eprint.iacr.org/2008/390, Citations in this document: §3
48. Koç, Ç.K., Paar, C. (eds.): CHES 2000. LNCS, vol. 1965. Springer, Heidelberg (2000); ISBN 3-540-42521-7, See [38]
49. Kurosawa, K. (ed.): ASIACRYPT 2007. LNCS, vol. 4833. Springer, Heidelberg (2007); ISBN 978-3-540-76899-9, See [14]
50. Matsui, M., Nakajima, J.: On the power of bitslice implementation on Intel Core2 processor. In: [56], pp. 121–134 (2007), Citations in this document: §1
51. Menezes, A., Qu, M.: Analysis of the Weil descent attack of Gaudry, Hess and Smart. In: [54], pp. 308–318 (2001), Citations in this document: §3
52. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation 48, 243–264 (1987), http://links.jstor.org/sici?sici=0025-5718(198701)48:177<243:STPAEC>2.0.CO;2-3; ISSN 0025-5718. MR 88e:11130, Citations in this document: §1, §3, §3, §3, §3
53. Montgomery, P.L.: Five, six, and seven-term Karatsuba-like formulae. IEEE Transactions on Computers 54, 362–369 (2005); Citations in this document: §2

54. Naccache, D. (ed.): CT-RSA 2008. LNCS, vol. 4964. Springer, Heidelberg (2008); ISBN 3-540-41898-9. MR 2003a:94039, See [51]
55. Oswald, E., Rohatgi, P. (eds.): CHES 2008. LNCS, vol. 5154. Springer, Heidelberg (2008); ISBN 978-3-540-85052-6, See [16], [36]
56. Paillier, P., Verbauwhede, I. (eds.): CHES 2007. LNCS, vol. 4727. Springer, Heidelberg (2007); ISBN 978-3-540-74734-5, See [50]
57. Peter, S., Langendörfer, P.: An efficient polynomial multiplier in GF(2^m) and its application to ECC designs. In: [5] (2007), `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=4211749&arnumber=4211979&count=305&index=229`, Citations in this document: §2
58. Qing, S., Okamoto, T., Zhou, J. (eds.): ICICS 2001. LNCS, vol. 2229. Springer, Heidelberg (2001); ISBN 3-540-42880-1, See [8]
59. Rodríguez-Henríquez, F., Koç, Ç.K.: On fully parallel Karatsuba multipliers for GF(2^m). In: [60], pp. 405–410 (2003); Citations in this document: §2, §2
60. Sahni, S. (ed.): Proceedings of the international conference on computer science and technology. Acta Press (2003); See [59]
61. Schönhage, A.: Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. Acta Informatica 7, 395–398 (1977), `http://cr.yp.to/bib/entries.html#1977/schoenhage`, ISSN 0001–5903. MR 55:9604, Citations in this document: §2
62. Schönhage, A., Strassen, V.: Schnelle Multiplikation großer Zahlen. Computing 7, 281–292 (1971), `http://cr.yp.to/bib/entries.html#1971/schoenhage-mult`, ISSN 0010–485X. MR 45:1431. Citations in this document: §2
63. Stein, W. (ed.): Sage Mathematics Software (Version 3.2.3) The Sage Group (2009), `http://www.sagemath.org`, Citations in this document: §1
64. Toom, A.L.: The complexity of a scheme of functional elements realizing the multiplication of integers. Soviet Mathematics Doklady 3, 714–716 (1963); ISSN 0197–6788. Citations in this document: §2
65. van der Poorten, A.J., Stein, A. (eds.): ANTS-VIII 2008. LNCS, vol. 5011. Springer, Heidelberg (2008); ISBN 978-3-540-79455-4, See [22]
66. Varadharajan, V., Mu, Y. (eds.): ACISP 2001. LNCS, vol. 2119. Springer, Heidelberg (2001); ISBN 978-3-540-42300-3, See [21]
67. von zur Gathen, J., Shokrollahi, J.: Fast arithmetic for polynomials over F_2 in hardware. In: [3], pp. 107–111 (2006); Citations in this document: §2, §2, §2
68. Weimerskirch, A., Paar, C.: Generalizations of the Karatsuba algorithm for efficient implementations (2006), `http://eprint.iacr.org/2006/224`, Citations in this document: §2
69. Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.): PKC 2006. LNCS, vol. 3958. Springer, Heidelberg (2006); ISBN 978-3-540-33851-2, See [12], [25]
70. Zhou, J., López, J., Deng, R.H., Bao, F. (eds.): ISC 2005. LNCS, vol. 3650. Springer, Heidelberg (2005); ISBN 3-540-29001-X, See [24]

Solving Hidden Number Problem with One Bit Oracle and Advice

Adi Akavia[⋆]

Institute for Advanced Study, Princeton NJ 08540
DIMACS, Rutgers University, Piscataway, NJ 08854

Abstract. In the *Hidden Number Problem (HNP)*, the goal is to find a hidden number s, when given p, g and access to an oracle that on query a returns the k most significant bits of $s \cdot g^a \bmod p$.

We present an algorithm solving HNP, when given an advice depending only on p and g; the running time and advice length are polynomial in $\log p$. This algorithm improves over prior HNP algorithms in achieving: (1) optimal number of bits $k \geq 1$ (compared with $k \geq \Omega(\log\log p)$); (2) robustness to random noise; and (3) handling a wide family of predicates on top of the most significant bit.

As a central tool we present an algorithm that, given oracle access to a function f over $\mathbb{Z}_N$, outputs all the significant Fourier coefficients of f (i.e., those occupying, say, at least 1% of the energy). This algorithm improves over prior works in being:

– *Local.* Its running time is polynomial in $\log N$ and $L_1(\widehat{f})$ (for $L_1(\widehat{f})$ the sum of f's Fourier coefficients, in absolute value).
– *Universal.* For any N, t, the *same* oracle queries are asked for *all* functions f over $\mathbb{Z}_N$ s.t. $L_1(\widehat{f}) \leq t$.
– *Robust.* The algorithm succeeds with high probability even if the oracle to f is corrupted by random noise.

1 Introduction

The *Hidden Number Problem (HNP)* was introduced by Boneh and Venkatesan [4] in the context of proving bit security for the Diffie-Hellman function. In HNP, for p a prime, and g a generator of $\mathbb{Z}_p^*$, the goal is to find a hidden number $s \in \mathbb{Z}_p^*$, when given p, g and oracle access to the function

$$P_{p,s,k}(a) \overset{def}{=} MSB_{p,k}(s \cdot g^a \bmod p)$$

mapping each $a \in 1, \ldots, p$ to the k most significant bits in the binary representation of $s \cdot g^a \bmod p$.

Boneh-Venkatesan [4] gave an algorithm solving HNP for any $k \geq \sqrt{\log p} + \log\log p$ in running time polynomial in $\log p$ (aka, *efficient*). Subsequently, Boneh-Venkatesan [5] gave an efficient algorithm solving HNP for $k \geq \Omega(\log\log p)$ provided the algorithm is given a short *advice* depending only on p and g (and not on s). Extensions to the case g is not a generator are given in [8,14,15].

⋆ This research was supported in part by NSF grant CCF-0514167, by NSF grant CCF-0832797, and by Israel Science Foundation 700/08. akavia@ias.edu

1.1 New Result: Solving HNP with One Bit Oracle and Advice

We present an efficient algorithm solving HNP for any $k \geq 1$, provided the algorithm is given a short advice depending only on p and g (and not on s).

Furthermore, our algorithm handles:

– *Random noise.* With high probability, our algorithm finds s even if the oracle answers are flipped independently at random with sufficiently small probability $\varepsilon > 0$. (Success probability is taken over the noise.)
– *Concentrated predicates.* Our algorithm finds s even when oracle access is to the function
$$P_{p,s}(a) \overset{def}{=} P_p(s \cdot g^a \bmod p)$$
where $\mathcal{P} = \{P_p\}$ is any family of "concentrated" predicates. We say that $\mathcal{P}$ is *concentrated* if
$$\exists c, \delta \text{ s.t. } \forall P_p \in \mathcal{P}, L_1(\widehat{P_p}) \leq (\log p)^c \text{ and } \mathsf{maj}(P_p) \leq 1 - \delta$$
for $L_1(\widehat{P_p}) \overset{def}{=} \sum_\alpha \left|\widehat{P_p}(\alpha)\right|$ the sum of Fourier coefficients, and $\mathsf{maj}(P_p) \overset{def}{=} \max_{b=0,1} \Pr_{a \in \mathbb{Z}_p}[P_p(a) = b]$ the frequency of the most common value.

Noise is tolerated up to $\varepsilon = c'\tau(\mathcal{P})$ for any $c' < 1$ and for any $\tau(\mathcal{P})$ a lower bound on the maximum squared magnitude of the (non-trivial) Fourier coefficients of predicates $P_p \in \mathcal{P}$. In particular, for $\mathcal{P}$ the most significant bit, $\varepsilon = O(1)$.[1]

As a corollary of our algorithm for HNP, we obtain bit security results for Diffie-Hellman related functions.

Our result improves on prior HNP algorithms (and the corresponding bit security results) in achieving:

1. Optimal number of bits $k \geq 1$ (rather than $k \geq \Omega(\log \log p)$);
2. Robustness to ε-random noise for substantial ε (e.g., ε is $O(1)$ rather than $O(1/\log p)$ for $\mathcal{P} = \mathcal{MSB}_k$ the k most significant bits); and
3. Handling the wide family of concentrated predicates (rather than only $\mathcal{MSB}_k$).

1.2 New Tool: Universally Finding Significant Fourier Coefficients

As a central tool we present an algorithm that finds the significant Fourier coefficients of a complex valued functions f over $\mathbb{Z}_p$, when given oracle access to f (aka, SFT algorithm).

Indexing Fourier coefficients by elements α in $\mathbb{Z}_p$, we say that α is *τ-significant* if its Fourier coefficient occupies at least τ-fraction of the energy
$$\left|\widehat{f}(\alpha)\right|^2 \geq \tau \sum_{\beta \in \mathbb{Z}_p} \left|\widehat{f}(\beta)\right|^2 .$$

[1] For $\mathcal{P}$ the $k \geq \Omega(\log \log p)$ most significant bits, prior works [5] tolerate adversarial noise corrupting up to $\varepsilon = O(1/\log p)$ fraction of the oracle values.

Our SFT algorithm, given p, τ, t, and oracle access to a function f over $\mathbb{Z}_p$ s.t. $L_1(\widehat{f}) \leq t$, outputs all the τ-significant Fourier coefficients of f. Our SFT algorithm is:

- *Local.* Its running time is polynomial in $\log p$, $1/\tau$ and t.
- *Universal.* For any p, τ and t, the *same* oracle queries are asked for *all* functions f over $\mathbb{Z}_p$ s.t. $L_1(\widehat{f}) \leq t$.
- *Robust.* With high probability, the algorithm succeeds even if the oracle to f is corrupted by random noise (probability is taken over the noise). Tolerated noise parameters are up to $\varepsilon = c\tau$ for any constant $c < 1$.

This improves over prior works in giving: (i) The first universal algorithm handling all functions f over $\mathbb{Z}_p$ (complexity scales with $L_1(\widehat{f})$). (ii) The first analysis proving robustness to noise in the context of universal SFT algorithms. We remark that these improvements are of independent interest in the context of sparse Fourier approximation, compressed sensing and sketching (cf. [3]).

Comparison to other SFT algorithms. For functions over the boolean hyper-cube $\mathbb{Z}_2^n$, Kushilevitz-Mansour (KM) gave a local universal SFT algorithm almost two decades ago [12]. Our algorithm matches the KM benchmark for the case of functions over $\mathbb{Z}_p$ for any positive integer p.

For functions over $\mathbb{Z}_p$, prior SFT algorithms [6,2,7] are not universal. In concurrent works [10,11] gave a universal SFT algorithm for a restricted class of functions over $\mathbb{Z}_p$: *compressible* or *Fourier sparse* functions.[2]

Noise is out of scope in the analysis of the universal algorithms [12,10,11].

These SFT algorithms [12,6,2,7,10,11] are insufficient for our result solving HNP. Both universality as well as handling functions that are neither compressible nor Fourier sparse are crucial for our algorithm solving HNP. Robustness to noise leads to robustness when solving HNP.

1.3 Techniques Overview

In HNP the goal is to find a hidden number s when given p, g and oracle access to a function $P_{p,s}$. We reduce the HNP problem to the problem of the finding significant Fourier coefficients of a function f_s defined by

$$f_s(y) \stackrel{def}{=} P_{p,s}(DL_{p,g}(y))$$

for $DL_{p,g}(y)$, the discrete log of y, i.e., the $a \in \mathbb{Z}_{p-1}$ s.t. $y = g^a \bmod p$. We then find the significant Fourier coefficients of f_s using our universal SFT algorithm.

Universality is crucial. Finding the Fourier coefficients of f_s requires access to f_s. To read the values $f_s(y)$ on entries y it suffices to query $P_{p,s}$ on the *discrete-logs* $DL_{p,g}(y)$. With universal algorithms, access to all entries y read

[2] For g a function over $\mathbb{Z}_p$ and $c, c' > 0$ absolute constants (indep. of p), g is *compressible* if for all i, the i-th largest Fourier coefficient of g has magnitude at most $O(1/c^i)$; and g is *Fourier sparse* if it has at most $(\log p)^{c'}$ non-zero Fourier coefficients.

by the algorithm can be granted using an *advice* depending only on p. This is because universal algorithms read a *fixed* set of entries y for all the considered functions over $\mathbb{Z}_p$; implying that the discrete-logs $DL_{p,g}(y)$ for all read entries y can be provided via an advice depending only on p. In contrast, with non-universal algorithms, providing access to f_s is intractable (assuming computing discrete logs is intractable).

Achieving universality. We say that a set of queries $S \subseteq \mathbb{Z}_p$ is *good* if we can find the significant Fourier coefficients of all considered function over $\mathbb{Z}_p$ when reading only entries in S. We present a combinatorial condition on sets S, and prove that any set S satisfying this condition is good. Furthermore, we show that sets S satisfying the condition exists, and can be efficiently construction by a randomized algorithm. We remark that explicit constructions of such good sets are given in subsequent works [3].

The combinatorial condition is that $S = \cup_{\ell=0}^{\log p}(A - B_\ell)$ for A a small biased set and B_ℓ's that are "small biased on $[0..2^\ell]$"; where we say that B *has small bias on* I if Fourier coefficients of (the characteristic function of) B approximate the Fourier coefficients of (the characteristic function) of I.

We prove that such sets S are good in two parts. First, for functions with bounded $L_1(\widehat{f})$, we prove S is good using Fourier analysis. Second, for noise corrupted functions $f' = f + \eta$, we prove S is good by showing the algorithm behaves similarly on the noisy and non-noisy functions. The latter is needed, as the Fourier approach fails for noisy f' due to their typically huge $L_1(\widehat{f'}) \approx \sqrt{p}$.

Comparison to prior works. Prior algorithms solving HNP follow a lattice based approach dating back to [4], in which HNP is reduced to the problem of finding closest lattice vectors (CVP), and the latter is solved using LLL algorithm [13]. In comparison, we take a Fourier approach inspired by [2].

We compare the set of queries used in the different SFT algorithms.

In the universal SFT algorithm for functions over the boolean hypercube $\mathbb{Z}_2^n$ [12], the set of queries is constructed using small biased sets in $\mathbb{Z}_2^n$, and the proof is Fourier analysis based.

In the (non-universal) SFT algorithms for functions over $\mathbb{Z}_p$ [6,2,7], the set of queries must be freshly chosen for each given input function f. Their analysis proves success with high probability over the sampled set of queries using deviation from expectation bounds.

In the universal SFT algorithm for (restricted class of) functions over $\mathbb{Z}_p$ [10,11], the set of queries is constructed using "K-majority k-strongly selective sets".

1.4 Paper Organization

The rest of this paper is organized as follows. In section 2 we summarize preliminary terminology, notations and facts. In section 3 we present our algorithm solving HNP with advice. In section 4 we present our universal SFT algorithm. In section 5 we discuss bit security implications.

2 Preliminaries

In this section we summarize preliminary terminology, notations and facts.

Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{C}$ denote the natural, integer, real and complex numbers respectively. Let $\mathbb{P}$ denote the set of all primes. Let $\mathbb{Z}_N$ and $\mathbb{Z}_N^*$ denote the additive and the multiplicative groups of integers modulo N. We identify the elements of $\mathbb{Z}_N$ with integers in $0, \ldots, N-1$, and denote $\mathsf{abs}(\alpha) = \min\{\alpha, N-\alpha\}$ for all $\alpha \in \mathbb{Z}_N$. Let $\mathbb{B}_r \overset{def}{=} \{z \in \mathbb{C} \mid |z| \leq r\}$ denote the complex ball of radius r.

2.1 Fourier Transform

We give definitions and properties for normed spaces and Fourier transform.

Inner product, norms, convolution. The *inner product* of complex valued functions f, g over a domain G is $\langle f, g\rangle \overset{def}{=} \frac{1}{|G|}\sum_{x\in G} f(x)\overline{g(x)}$. Denote the normalized ℓ_2 norm of f by $\|f\|_2 \overset{def}{=} \sqrt{\langle f, f\rangle}$, its ℓ_∞ norm by $\|f\|_\infty \overset{def}{=} \max\{|f(x)| \mid x \in G\}$, and its un-normalized L_1-norm by $L_1(f) \overset{def}{=} \sum_{x\in G}|f(x)|$. The *convolution* of f and g is the function $f * g\colon G \to \mathbb{C}$ defined by $f * g(x) \overset{def}{=} \frac{1}{|G|}\sum_{y\in G} f(y)\overline{g(x-y)}$.

Characters and Fourier transform. The *characters* of $\mathbb{Z}_N$ are the functions $\chi_\alpha\colon \mathbb{Z}_N \to \mathbb{C}$, $\alpha \in \mathbb{Z}_N$, defined by $\chi_\alpha(x) \overset{def}{=} e^{2\pi i\alpha x/N}$. The *Fourier transform* of a complex valued function f over $\mathbb{Z}_N$ is the function $\widehat{f}\colon \mathbb{Z}_N \to \mathbb{C}$ defined by $\widehat{f}(\alpha) \overset{def}{=} \langle f, \chi_\alpha\rangle$. For any $\alpha \in \mathbb{Z}_N$ and $\tau \in [0,1]$, we say that α is a τ-*significant* Fourier coefficient iff $\left|\widehat{f}(\alpha)\right|^2 \geq \tau\|f\|_2^2$. Denote by $\mathsf{Heavy}_\tau(f)$ the set of all τ-significant Fourier coefficients of f.

A few useful properties of the Fourier transform follow.

Proposition 1. *For any $f, g\colon \mathbb{Z}_N \to \mathbb{C}$,*

1. *Parseval Identity:* $\frac{1}{N}\sum_{x\in\mathbb{Z}_N}|f(x)|^2 = \sum_\alpha \left|\widehat{f}(\alpha)\right|^2$.
2. *Convolution Theorem:* $\widehat{(f * g)}(\alpha) = \widehat{f}(\alpha)\cdot\widehat{g}(\alpha)$.
3. *Phase Shift: For any $\alpha_0 \in \mathbb{Z}_N$, if $g = f\cdot\chi_{-\alpha_0}$, then $\widehat{g}(\alpha) = \widehat{f}(\alpha-\alpha_0)$ (where subtraction is modulo N).*
4. *Scaling: For any $s \in \mathbb{Z}_N^*$, if $g(x) = f(sx)\ \forall x$, then $\widehat{g}(\alpha) = \widehat{f}(\alpha\cdot s^{-1})\ \forall\alpha$ (where multiplication and inverse are modulo N).*

Proof. Proof is standard, see [16]. □

Proposition 2. *Let $S_t(\alpha) \overset{def}{=} \frac{1}{t}\sum_{y=0}^{t-1}\chi_\alpha(y)$ for some $t \in [0..N-1]$. Then:*

1. $|S_t(\alpha)|^2 = \frac{1}{t^2}\frac{1-\cos(\frac{2\pi}{N}\alpha t)}{1-\cos(\frac{2\pi}{N}\alpha)}$
2. *Pass Band:* $\forall \alpha \in \mathbb{Z}_N$ *and* $\gamma \in [0,1]$, *if* $\mathsf{abs}(\alpha) \le \gamma\frac{N}{2t}$, *then* $|S_t(\alpha)|^2 > 1 - \frac{5}{6}\gamma^2$
3. *Fast decreasing:* $\forall \alpha \in \mathbb{Z}_N$, $|S_t(\alpha)|^2 < \frac{2}{3}\left(\frac{N/t}{\mathsf{abs}(\alpha)}\right)^2$
4. *Fourier bounded:* $\forall \alpha \in \mathbb{Z}_N$, $|S_t(\alpha)|^2 \le 1$

Proof. Recall that $\chi_\alpha(x) = \omega^{\alpha x}$ for $\omega = e^{i\frac{2\pi}{N}}$ a primitive root of unity of order N. By the formula for geometric sum $S_t(\alpha) = \frac{1}{t}\frac{\omega^{-\alpha t}-1}{\omega^{-\alpha}-1}$. Assigning $w^\beta = \cos(2\pi\beta/N) + i\sin(2\pi\beta/N)$ for $\beta = \alpha t$ in the numerator and $\beta = \alpha$ in the denominator and using standard trigonometric identities, we conclude that $|S_t(\alpha)|^2 = \frac{1}{t^2}\frac{1-\cos(\frac{2\pi}{N}\alpha t)}{1-\cos(\frac{2\pi}{N}\alpha)}$. The upper and lower bounds on S_t are obtained using the Taylor approximation for the cosine function: $1 - \frac{\theta^2}{2!} \le \cos(\theta) \le 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!}$. Details appear in [2,1]. $\square$

2.2 Chernoff/Hoeffding Tail Inequality

The Chernoff/Hoeffding bound on the deviation from expectation of sums of independent random variables follows.

Proposition 3 (Chernoff/Hoeffding Bound [9]). *Let $X_1, \ldots, X_t$ be independent random variables of expectations $\mu_1, \ldots, \mu_y$ and bounded values $|X_i| \le M$. Then, $\forall \eta > 0$, $\Pr[\left|\frac{1}{t}\sum_{i=1}^t X_i - \frac{1}{t}\sum_{i=1}^t \mu_i\right| \ge \eta] \le 2 \cdot exp\left(-\frac{2t\eta^2}{M^2}\right)$.*

2.3 Noise Models

We say that η is an *ε-random noise* if its values $\eta(a)$, $a \in \mathbb{Z}_p$, are chosen independently at random from distributions of expected absolute values at most $\mathbb{E}[|\eta(a)|] \le \varepsilon$.

We focus on additive noise η corrupting functions f to a function $f' = f + \eta$. Without loss of generality, f and η accept values in the balls $\mathbb{B}_1$, $\mathbb{B}_2$ respectively.

3 Solving Hidden Number Problem with Advice

In this section we present our algorithm solving with advice $\mathrm{HNP}^{\mathcal{P},\varepsilon}$.

Fix a family of functions $\mathcal{P} = \left\{P_p\colon \mathbb{Z}_p^* \to \mathbb{B}_1\right\}_{p\in\mathbb{P}}$ and a noise parameter ε.

Definition 1 (Hidden Number Problem). *In the* (extended) Hidden Number Problem $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ *the goal is to find a hidden number $s \in \mathbb{Z}_p^*$, when given a prime p, a generator g of $\mathbb{Z}_p^*$, and oracle access to the function*

$$P'_{p,s}(a) \overset{def}{=} P_p(s \cdot g^a \bmod p) + \eta(a)$$

for η an ε-random noise.

Let ℓ, q, t be functions over $\mathbb{P}$. We say that an algorithm (ℓ, q, t)-*solves* $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ if there is an advice $Adv_{p,g}$ depending only on p, g of length $|Adv_{p,g}| \leq \ell(p)$, such that the following holds. Given p, g, $Adv_{p,g}$, and oracle access to $P'_{p,s}$, the algorithm outputs s with probability at least $q(p)$; and its running time is at most $t(p)$. We say that the algorithms *solves with advice* $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ if $1/q(p), \ell(p)$ and $t(p)$ are polynomial in $\log p$.

3.1 Solving with Advice $\mathrm{HNP}^{\mathcal{P},\varepsilon}$: Concentrated $\mathcal{P}$

We present an efficient algorithm solving with advice $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ for concentrated $\mathcal{P}$. We remark that concentration defined here differ than concentration in [2].

Let M, τ and α be functions mapping indices $p \in \mathbb{P}$ into non-negative reals $M(p), \tau(p)$ and a non-zero element $\alpha(p) \in \mathbb{Z}_p$.

Definition 2 (Concentration). *$\mathcal{P}$ is (M, τ, α)-*concentrated *if for all $P_p \in \mathcal{P}$,*

$$L_1(\widehat{P_p}) \leq M(p) \quad \textit{and} \quad \left|\widehat{P_p}(\alpha(p))\right|^2 \geq \tau(p).$$

$\mathcal{P}$ is concentrated *if $\exists c > 0$ s.t. $\forall p \in \mathbb{P}$, $M(p)$ and $1/\tau(p)$ are at most $(\log p)^c$.*

Let $\tau(\mathcal{P})$ denote a lower bound on the maximum weight $\left|\widehat{P_p}(\alpha)\right|^2$ of non-trivial Fourier coefficients $\alpha \neq 0$, for all $P_p \in \mathcal{P}$.

Theorem 1 ($\mathrm{HNP}^{\mathcal{P},\varepsilon}$). *For any concentrated $\mathcal{P}$ and $\varepsilon \leq c \cdot \tau(\mathcal{P})$ for $c < 1$, there exists an algorithm that solves with advice $HNP^{\mathcal{P},\varepsilon}$.*

Proof. Let m, τ, α be s.t. $\mathcal{P}$ is (M, τ, α)-concentrated. We present an algorithm that (ℓ, q, t)-solves $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ for $q(p) \geq \Omega(\tau(p))$ and for $\ell(p), t(p)$ polynomial in $\log p$, $M(p)$ and $1/\tau(p)$. The advice we use is:

$$Adv_{p,g} \overset{def}{=} \{(x, DL_{p,g}(x))\}_{x \in S}$$

for $S \subseteq \mathbb{Z}_p$ a set of good queries for our universal SFT algorithm on input parameters p, $\tau(p)$ and $M(p)$ (cf. Definition 4). The function $f_s = f_{p,g,s}$ over $\mathbb{Z}_p$ is defined by

$$f_s(x) \overset{def}{=} P'_{p,s}(DL_{p,g}(x))$$

for all $x \in \mathbb{Z}_p^*$ and $f_s(0) = 0$. Note that we can access $f_s(x)$ for all $x \in S$ by querying $P'_{p,s}$ on $a = DL_{p,g}(x)$ provided in the advice. Our algorithm for $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ follows.

```
Algorithm 1 Solving HNP$^{\mathcal{P},\varepsilon}$.

1. Run the SFT Algorithm 2 on input $p$,$\tau(p)$,$M(p)$, and oracle
   access to the restriction of $f_s$ to $S$; denote its output by $L$.
2. Output $((\alpha(p))^{-1} \cdot \beta)^{-1}$ for a uniformly random $\beta \in L$.
```

We show that Algorithm 1 outputs the hidden number s with probability $q(p) \geq \Omega(\tau(p))$. Fix p and denote $\alpha = \alpha(p)$, $\tau = \tau(p)$. Recall that $\left|\widehat{P_p}(\alpha)\right|^2 \geq \tau$ (since $\mathcal{P}$ is (M, τ, α)-concentrated), and that $\widehat{P_{p,s}}(\beta) = \widehat{P_p}(\beta s^{-1})$ $\forall \beta$ (by Proposition 1 Item 4 and the definition of $P_{p,s}(x) = P_p(s \cdot x)$). Therefore, the αs^{-1}-Fourier coefficient of $P_{p,s}$ is τ-significant, i.e.,

$$\left|\widehat{P_{p,s}}(\alpha \cdot s^{-1})\right|^2 \geq \tau.$$

Thus $L \ni \alpha s^{-1}$ with probability at least $1 - 1/p^{\Omega(1)}$ (by Theorem 4). Implying that

$$\beta = \alpha s^{-1}$$

with probability at least $(1 - 1/p^{\Omega(1)})/\left|L\right| \geq \Omega(\tau)$ (since β is a random element in L, and employing the bound $|L| \leq O(1/\tau)$ from Theorem 4). When $\beta = \alpha s^{-1}$, the output is

$$(\alpha^{-1}\beta)^{-1} = (\alpha^{-1}(\alpha s^{-1}))^{-1} = s.$$

We conclude that the output is s with probability $q(p) \geq \Omega(\tau)$.

Finally, the advice length $\ell(p)$ and the running time $t(p)$ are dominated by the query complexity and running time of the SFT Algorithm which is polynomial in $\log p$, $1/\tau(p)$ and $M(p)$ (cf. Theorem 4). □

Remark 1. Tighter bounds on the success probability $q(p)$ are possible at times. E.g., for the most significant bits $\mathcal{P} = \mathcal{MSB}_k$ for any $k \geq 1$, $q(p) \geq 1/2$.

3.2 Solving with Advice HNP$^{\mathcal{P},\varepsilon}$: Segment Predicates $\mathcal{P}$

We solve with advice HNP$^{\mathcal{P},\varepsilon}$ for segment predicates $\mathcal{P}$.

Let $\mathcal{P} = \left\{P_p \colon \mathbb{Z}_p^* \to \{\pm 1\}\right\}_{p \in \mathbb{P}}$. Let σ, a be functions mapping primes p to positive integers $\sigma(p)$ and to elements $a(p) \in \mathbb{Z}_p^*$. Denote by $\sigma(\mathcal{P})$ an upper bound on $\sigma(p)$ for all p.

Definition 3 (Segment Predicates [2]). *$\mathcal{P}$ is a (σ, a)-*segment predicate *if $\forall p$, $\exists P'_p \colon \mathbb{Z}_p^* \to \{\pm 1\}$ s.t.*

– *$P_p(x) = P'_p(x \cdot a(p))$ for all x, and*
– *$P'_p(x+1) \neq P'_p(x)$ for at most $\sigma(p)$ x's in $\mathbb{Z}_p$.*

$\mathcal{P}$ is a segment predicate *if $\exists c > 0$ s.t. $\sigma(p) < (\log p)^c$ for all p.*

We say that $\mathcal{P}$ is *far from constant* if $\exists \delta > 0$ s.t. $\forall p$, $\mathsf{maj}(P_p) \leq 1 - \delta$ for $\mathsf{maj}(P_p)$ the frequency of P_p's most common value.

Theorem 2. *Let $\mathcal{P}$ be a far from constant segment predicate and $\varepsilon \leq c/\sigma(\mathcal{P})$ for $c < 1$. Then there exists an algorithm that solves with advice HNP$^{\mathcal{P},\varepsilon}$.*

Proof. By Lemma 1, if $\mathcal{P}$ is a segment predicate, then $\mathcal{P}$ is concentrated; and furthermore, $\tau(\mathcal{P}) \geq 1/\sigma(\mathcal{P})$. By Theorem 1 this implies that there exists an algorithm that solves with advice $\mathrm{HNP}^{\mathcal{P},\varepsilon}$. □

Lemma 1. *If $\mathcal{P}$ is a (σ, a)-segment predicate, then $\mathcal{P}$ is (M, τ, α)-concentrated for $M(p) = O(\sigma(p) \ln p)$, $\tau(p) = \Omega(1/\sigma(p))$, and $\alpha(p) = a(p)$.*

Proof. For each $P_p \in \mathcal{P}$, extend P_p to a function over $\mathbb{Z}_p$ by setting $P_p(0) = P_p(1)$. Fix p and drop its indices.

Consider first the case $a(p) = 1$. To show that $L_1(\widehat{P}) \leq M(p)$ and $\widehat{P}(1) \geq \tau(p)$, we first show that $\widehat{P}(\alpha) = \sum_{j=1}^{\sigma+1}(\ell_j/p)S_{\ell_j}(\alpha)$ for all $\alpha \in \mathbb{Z}_p$. A segment predicate with $a = 1$ defines a partition of $\mathbb{Z}_p$ into $\sigma + 1$ segments I_j, so that P is a constant $b_j \in \{\pm 1\}$ on each segment I_j. Thus, we can express P as a sum, $P = \sum_{j=1}^{\sigma+1} P_j$, of functions $P_j \colon \mathbb{Z}_p \to \{-1, 0, 1\}$ such that $P_j(x)$ is the constant $P(x)$ for $x \in I_j$ and 0 otherwise. By the linearity of the Fourier transform, for all $\alpha \in \mathbb{Z}_p$, $\widehat{P}(\alpha) = \sum_{j=1}^{\sigma+1} \widehat{P_j}(\alpha)$. By definition of the Fourier transform, $\widehat{P_j}(\alpha) = \frac{1}{p}\sum_{x \in I_j} b_j \chi_\alpha(x)$. Thus for c_j the starting point of I_j and $\ell_j = |I_j|$ its length, $\left|\widehat{P_j}(\alpha)\right| = |\chi_\alpha(c_j)| \left|\frac{1}{p}\sum_{x=0}^{\ell_j} \chi_\alpha(x)\right| = (\ell_j/p)S_{\ell_j}(\alpha)$ for $S_{\ell_j}(\alpha) = \frac{1}{\ell_j}\sum_{x=0}^{\ell_j - 1}\chi_\alpha(x)$ as defined in Proposition 2. We conclude that $\left|\widehat{P}(\alpha)\right| = \sum_{j=1}^{\sigma+1}(\ell_j/p)S_{\ell_j}(\alpha)$.

We show that $L_1(\widehat{P}) \leq O(\sigma \ln p)$. By Proposition 2, $\left|S_{\ell_j}(\alpha)\right| \leq O\left(\frac{p/\ell_j}{\mathsf{abs}(\alpha)}\right)$ for all ℓ_j, implying that $\left|\widehat{P}(\alpha)\right| \leq \sum_{j=1}^{\sigma+1} O\left(\frac{(\ell_j/p)(p/\ell_j)}{\mathsf{abs}(\alpha)}\right) = O\left(\sigma/\mathsf{abs}(\alpha)\right)$. Thus, $L_1(\widehat{P}) = \sum_\alpha \left|\widehat{P}(\alpha)\right| \leq O\left(\sigma \cdot \sum_\alpha \frac{1}{\mathsf{abs}(\alpha)}\right) = O(\sigma \ln p)$.

We show that $\left|\widehat{P}(1)\right| \geq \Omega(1/\sigma)$. Let ℓ_{j^*} be the length of the second longest segment in $I_1, \ldots, I_{\sigma+1}$. Clearly $\ell_{j^*} \leq p/2$. Moreover, $\ell_{j^*} \geq \Omega(p/\sigma)$ because for far from constant $\mathcal{P}$, the longest segment is of length at most $(1-c)p$ for $c > 0$, implying that the second longest is of length at least the average length cp/σ over the remaining σ segments. By Proposition 2, $|S_\ell(1)|^2 \geq \Omega(1)$ for all $\ell \leq p/2$. Thus, $\left|\widehat{P}_{j^*}(\alpha)\right|^2 \geq (\ell_{j^*}/p) \cdot \Omega(1) = \Omega(1/\sigma)$. We conclude that for $\alpha(p) = 1$ there is a function $\tau(p) \geq \Omega(1/\sigma(p))$ such that $\left|\widehat{P}(\alpha(p))\right|^2 \geq \tau(p)$ for all $p \in \mathbb{P}$.

Consider next the case of $a(p) \neq 1$. By definition of segment predicates, there exists P' s.t. $P(x) = P'(xa)$ for all $x \in \mathbb{Z}_p^*$. Extend P' to $\mathbb{Z}_p$. By Proposition 1, for all $\alpha \in \mathbb{Z}_p$ $\widehat{P(\alpha)} = \widehat{P}'(\alpha \cdot a^{-1})$. Implying that $L_1(\widehat{P}) = L_1(\widehat{P'}) \leq O(\sigma \ln p)$ (because $\left\{\alpha a^{-1}\right\}_{\alpha \in \mathbb{Z}_p} = \mathbb{Z}_p$ for any a co-prime to p), and $\widehat{P}(a) = \widehat{P'}(a \cdot a^{-1}) = \widehat{P'}(1) \geq \Omega(1/\sigma)$.

We conclude that any family $\mathcal{P}$ of (σ, a)-segment predicates is (M, τ, α)-concentrated for $M(p) \leq O(\sigma(p) \ln p)$, $\tau(p) \geq \Omega(1/\sigma(p))$ and $\alpha(p) = a(p)$. □

3.3 Solving with Advice $\mathrm{HNP}^{\mathcal{P},\varepsilon}$: The Single Most Significant Bit

We solve with advice $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ for $\mathcal{P} = \mathcal{MSB}$ the *single* most significant bit.

Let $\mathcal{MSB} = \left\{MSB_p\colon \mathbb{Z}_p^* \to \{\pm 1\}\right\}_{p\in\mathbb{P}}$ the family of predicates giving the *single* most significant bit $MSB_p(x)$ of x (in a ± 1 binary representation).

Theorem 3. *For any $\varepsilon = O(1)$ sufficiently small, there exists an algorithm that solves with advice HNP$^{\mathcal{MSB},\varepsilon}$.*

Proof. For the most significant bit MSB_p, $MSB_p(x+1) \neq MSB_p(x)$ only for one $x \in \mathbb{Z}_p^*$. Namely, $\mathcal{MSB}$ is a family of (σ, a)-segment predicates with $\sigma(p) = 1$, $a(p) = 1$ for all p. By Theorem 2, this implies that for any $\varepsilon = O(1)$ sufficiently small, there exists an algorithm that solves with advice HNP$^{\mathcal{P},\varepsilon}$. □

4 Universally Finding Significant Fourier Coefficients

In this section we present our universal SFT algorithm.

In the following We present the combinatorial condition on good queries sets S; show such sets exists; and prove that our SFT algorithm succeeds even when given oracle access only to the restriction of the input function f to the entries in S.

We define good queries. Recall that $A \subseteq \mathbb{Z}_N$ is γ*-biased* if $|\mathbb{E}_{x\in A}[\chi(x)]| < \gamma$ for all non-trivial characters χ of $\mathbb{Z}_N$. For $B, I \subseteq \mathbb{Z}_N$, we say that B is (γ, I)*-biased* if $|\mathbb{E}_{x\in B}[\chi(x)] - \mathbb{E}_{x\in I}[\chi(x)]| \leq \gamma$ for all characters χ is $\mathbb{Z}_N$. Denote by $A - B$ the set of differences $\{a - b\}_{a\in A, b\in B}$.

Definition 4 (Good Queries). *Let $\mathcal{S} = \{S_{N,\tau,t}\}_{N,\tau,t}$ be a family of sets $S_{N,\tau,t} \subseteq \mathbb{Z}_N$. We say that $\mathcal{S}$ is* good *if for all N, τ, t and for $\gamma = O(\tau/(t^2 \log N))$ sufficiently small, $S_{N,\tau,t} = \bigcup_{\ell=1}^{\lfloor(\log N)\rfloor}(A - B_\ell)$ s.t.*

- *A is γ-biased in $\mathbb{Z}_N$, of size $|A| = \Theta(\frac{1}{\gamma^2}\log N)$.*
- *$\forall \ell$, B_ℓ is $(\gamma, [0..2^\ell])$-biased in $\mathbb{Z}_N$, of size polynomial in $\log N$ and $1/\gamma$,*

We remark that the meaning of "sufficiently small γ" depends on the considered noise parameter ε, specifically, on the ratio $\varepsilon : \tau$. To simplify parameters, we fix this ratio to be, say, $\varepsilon < 0.9\tau$.

We show that good queries $\mathcal{S}$ exist. Moreover, there is a randomized algorithm that constructs good sets $S_{N,\tau,t}$ with high probability.

Proposition 4 (Good Queries Exist). *There is a randomized algorithm that given N, τ and t, outputs $S = S_{N,\tau,t}$ such that S is good with probability at least $1 - 1/N^{\Omega(1)}$; and its running time is $O(|S|)$.*

Proof. The algorithm outputs $S = \cup_{\ell=1}^{\lfloor(\log N)\rfloor}(A - B_\ell)$ for independent uniformly random sets $A \subseteq \mathbb{Z}_N$ and $B_\ell \subseteq [0..2^\ell]$, of sizes $|A| = O(\frac{1}{\gamma^2}\log N)$ and $|B_\ell| = O(\frac{1}{\gamma^2}\cdot \log N \cdot \log\log N)$, $\ell = 1, \ldots, \lfloor(\log N)\rfloor$.

Using Chernoff and Union bounds it is straightforward to show that S is good with probability at least $1 - 1/N^{\Omega(1)}$; details omitted. □

We show that our SFT algorithm succeeds when given oracle access to the restriction of the input function f (or its corruption by noise $f' = f + \eta$) to good queries $S = S_{N,\tau,t}$. Denote this restriction by $f'_{|S} \overset{def}{=} \{(x, f'(x))\}_{x \in S}$.

Let $\mathcal{S} = \{S_{N,\tau,t}\}$ be any family of good queries. For any integer $N > 0$, reals $\tau, t > 0$, a function $f\colon \mathbb{Z}_N \to \mathbb{B}_1$ s.t. $L_1(\widehat{f}) \leq t$, and an ε-random noise η for $\varepsilon < 0.9\tau$ the following holds.

Theorem 4 (SFT). *Our SFT algorithm, when given N, τ, t and $f'_{|S_{N,\tau,t}}$ for $f' = f + \eta$, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ of size $|L| \leq O(1/\tau)$, with probability at least $1 - 1/N^{\Omega(1)}$; and its running time is polynomial in $\log N$, $1/\tau$ and t.*

The probability is taken over the random noise η. In particular, when there is no noise, the success probability is 1.

Remark 2. Our SFT algorithm also handles: (i) Small amount of *adversarial noise*, that is, noise corrupting ε-fraction of the values of $f_{|S_{N,\tau,t}}$ for sufficiently small $\varepsilon = O(\tau/\log N)$. (ii) Input functions f accepting arbitrary complex values (and their corruption by noise f').

To prove Theorem 4, we first present the details of our SFT algorithm (Sect. 4.1), and then present its analysis (Sect. 4.2).

4.1 The SFT Algorithm

We give the details of our SFT algorithm. At a high level, the SFT algorithm is a binary search algorithm that repeatedly:

1. **Partitions** the set of potentially significant Fourier coefficients into two halves.
2. **Tests** each half to decide if it (potentially) contains a significant Fourier coefficient. This is done by **estimating** whether the sum of squared Fourier coefficients in each half exceeds the significance threshold τ.
3. **Continues recursively** on any half found to (potentially) contain significant Fourier coefficients.

At each step of this search, the set of potentially significant Fourier coefficients is maintained as a collection $\mathcal{J}$ of intervals: At the first step of the search, all Fourier coefficients are potentially significant, so $\mathcal{J}$ contains the single interval $J = [1..N]$. At each following search step, every interval $J \in \mathcal{J}$ is partitioned into two sub-intervals J_1 and J_2 containing the lower and upper halves of J respectively, and the set $\mathcal{J}$ is updated to hold only the sub-intervals that pass the test, i.e., those that (potentially) contain a significant Fourier coefficient. After $\log N$ steps this search terminates with a collection $\mathcal{J}$ of length one intervals revealing the frequencies of the significant Fourier coefficients. For all frequencies α of the significant Fourier coefficients, we then compute as an $O(\tau)$-approximation for $\widehat{f}(\alpha)$ the value $val_\alpha = \frac{1}{|A|}\sum_{x \in A-y} f(x)\overline{\chi_\alpha(x)}$ for some arbitrary $y \in \cup_{\ell=1}^{\lfloor(\log N)\rfloor} B_\ell$.

The heart of the algorithm is the test deciding which intervals potentially contain a significant Fourier coefficient (aka, distinguishing procedure). The distinguishing procedure we present, given an interval J, answers YES if its Fourier weight $weight(J) = \sum_{\alpha \in J} \left|\widehat{f}(\alpha)\right|^2$ exceed the significance threshold τ, and answers NO if the Fourier weight of a slightly larger interval $J' \supseteq J$ is less than $\tau/2$. This is achieved by estimating the ℓ_2 norm (i.e., sum of squared Fourier coefficients) of a filtered version of the input function f, when using a filter h that passes Fourier coefficients in J and decays fast outside of J.

The filters h that we use for depth ℓ of the search are the (normalized) *periodic square function* of support size 2^ℓ or Fourier domain translations of this function:

$$h_{\ell,c}(y) \overset{def}{=} \begin{cases} \frac{N}{2^\ell} \cdot \chi_{-c}(y) \; y \in [0..2^\ell] \\ 0 \qquad\qquad otherwise \end{cases} \tag{1}$$

The filter $h = h_{\ell,c}$ passes all frequencies that lie within the length $N/2^\ell$ interval J centered around c, and decays fast outside of J. The filtered version of f is $f * h$, and we estimate its ℓ_2 norm $\|f * h\|_2^2$ by the estimator:

$$\mathsf{est}_{\ell,c}(f) \overset{def}{=} \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c}(y) \overline{f(x-y)} \right)^2 \tag{2}$$

for $A, B_1, \ldots, B_\ell \subseteq \mathbb{Z}_N$ as specified in the definition of good queries 4.

A pseudo-code of the algorithm follows. We denote intervals by the pair $\{a, b\}$ of their endpoints. To simplify notations, we assume: $(a' + b')/2$ is an integer (otherwise, appropriate flooring/ceiling is taken); $\|f\|_2 = 1$ (otherwise we normalize f it by dividing each read value by an energy estimator $\frac{1}{|A|} \sum_{x \in A} f(x)^2$); $0 \in \bigcup_\ell B_\ell$ (otherwise we change variable in $\sum_{x \in A} \chi_\alpha(x) f(x)$ to $z = x - y$ for a random $y \in \bigcup_\ell B_\ell$).

```
Algorithm 2 SFT.
Input: $N \in \mathbb{N}$, $\tau \in (0,1]$, $\{(x, y, f(x-y))\}_{x \in A, y \in B_\ell}$ $\forall \ell = 1, \ldots, \lfloor(\log N)\rfloor$

1. Initialize: $\mathcal{J} \leftarrow \{\{0, N\}\}$
2. While $\exists \{a, b\} \in \mathcal{J}$ s.t. $b - a > 0$ do:
   (a) Delete $\{a, b\}$ from $\mathcal{J}$
   (b) For each pair $\{a', b'\}$ in $Low = \{a, \frac{a+b}{2}\}$, $High = \{\frac{a+b}{2} + 1, b\}$ do:
       i. Compute $\mathsf{est}_{\ell,c} \leftarrow \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c}(y) f(x-y)\right)^2$ for $\ell = \log(N/(b' - a'))$, $c = \lfloor((a' + b')/2)\rfloor$
      ii. If $\mathsf{est}_{\ell,c} \geq \tau/2$, insert $\{a', b'\}$ to $\mathcal{J}$
3. Sieving: For each $\{\alpha, \alpha\} \in \mathcal{J}$,
   (a) Compute $val(\alpha) \leftarrow \left|\frac{1}{|A|} \sum_{x \in A} \chi_\alpha(x) f(x)\right|^2$
   (b) If $val(\alpha) < \tau/2$, delete $\{\alpha, \alpha\}$ from $\mathcal{J}$
4. Output $L = \{\alpha \mid \{\alpha, \alpha\} \in \mathcal{J}\}$
```

4.2 Proof of Theorem 4

In this section we bring the proof of Theorem 4.

Proof of Theorem 4. Let $h_{\ell,c}$ and $\mathsf{est}_{\ell,c}(f)$ be as defined in (1)-(2). Fix a sufficiently small absolute constant $c > 0$. Consider condition (*) on $f' = f + \eta$:

$$(*) \quad \left|\mathsf{est}_{\ell,c}(f') - \|f * h_{\ell,c}\|_2^2\right| < c\tau \text{ for all } \ell = 1, \dots, \lfloor(\log N)\rfloor, c \in \mathbb{Z}_N$$

By Lemma 2, when (*) holds, the SFT algorithm outputs $L \supseteq \mathsf{Heavy}_\tau(f)$ in running time polynomial in $\log N$, $1/\tau$ and t. By Lemma 3, when S is a good, (*) holds with probability at least $1 - 1/N^{\Omega(1)}$ over the noise η. Thus, the SFT algorithm outputs $L \supseteq \mathsf{Heavy}_\tau(f)$ in time polynomial in $\log N$, $1/\tau$ and t.

Proving $|L| \le O(1/\tau)$ is similar. Consider condition $(*')$ saying that $\forall \alpha \in \mathbb{Z}_N$, $\left|\frac{1}{|A|}\sum_{x\in A} f'(x)\overline{\chi_\alpha(x)} - \widehat{f}(\alpha)\right| < c\tau$. We show that first, if (*') holds, then the sieving step leaves in $\mathcal{J}$ only $\{\alpha, \alpha\}$ s.t. $\left|\widehat{f}(\alpha)\right|^2 \ge \Omega(\tau)$; implying $|L| \le O(1/\tau)$ by Parseval Identity. Second, when S is good, (*') holds with high probability over the noise η. We conclude that $|L| \le O(1/\tau)$ with high probability over the noise η. Details omitted from this extended abstract. □

We show that the SFT algorithm succeed on functions f' satisfying (*).

Lemma 2. *Let $f' = f + \eta$ and all other parameters be as in Theorem 4. If conditions (*) holds for f', then the SFT algorithm returns a list $L \supseteq \mathsf{Heavy}_\tau(f)$ in running time polynomial in $\log N$, $1/\tau$ and t.*

Proof. Denote $J = [a', b']$, $\ell = \log(N/(b' - a'))$ and $c = (a' + b')/2$.

Correctness. Consider a significant Fourier coefficient $\alpha \in \mathbb{Z}_N$. To show that $\alpha \in L$, it suffices to show that $\mathsf{est}_{\ell,c}(f') > \tau/2$ whenever $J \ni \alpha$. The latter is true because when J contains a τ-significant Fourier coefficient, then by Proposition 21 Item (1), $\|f * h_{\ell,c}\|_2^2 \ge \Omega(\sum_{\alpha\in J}\left|\widehat{f}(\alpha)\right|^2) \ge \Omega(\tau)$, which by (*) implies that $\mathsf{est}_{\ell,c}(f') \ge \Omega(\tau) \ge \tau/2$ (the latter holds by setting appropriate constants).

Efficiency. Fix ℓ, to bound the running time it suffices to show that "$\mathsf{est}_{\ell,c}(f') \ge \tau/2$" does not happen for too many disjoint intervals J of length $N/2^\ell$. If $\mathsf{est}_{\ell,c}(f') \ge \tau/2$, then by condition (*), $\|h_{\ell,c} * f\|_2^2 \ge \Omega(\tau)$. By Claim 21 Item 2, the latter implies that for a slightly larger interval $J' \supseteq J$, $|J'|/|J| \le O(1/\gamma)$, its Fourier weight (that is, sum of squared Fourier coefficients with frequencies in J') is greater than $\Omega(\tau)$. This implies that $\mathsf{est}_{\ell,c}$ cannot be greater than $\tau/2$ too often, because there are at most $O(1/\tau)$ disjoint intervals whose Fourier weight exceeds $\Omega(\tau)$ (by Parseval Identity), and thus at most $O(\frac{1}{\tau} \cdot \frac{|J'|}{|J|})$ (possibly, overlapping) intervals J' whose Fourier weight exceeds $\Omega(\tau)$. □

Claim 21. *For integers $\ell, c > 0$ and real $\gamma > 0$, let $J_{\ell,c} = \left\{\alpha \mid \mathsf{abs}(\alpha - c) \le \frac{N}{2^\ell}\right\}$ an interval, and $J'_{\ell,c,\gamma} = \left\{\alpha \mid \mathsf{abs}(\alpha - c) \le \sqrt{\frac{2}{3\gamma}} \cdot \frac{N}{2^\ell}\right\}$ its extension. Then: (1) $\|h_{\ell,c} * f\|_2^2 \ge \frac{1}{6}\sum_{\alpha\in J_{\ell,c}}\left|\widehat{f}(\alpha)\right|^2$, and (2) $\|h_{\ell,c} * f\|_2^2 \le \sum_{\alpha\in J'_{\ell,c,\gamma}}\left|\widehat{f}(\alpha)\right|^2 + \gamma$.*

Proof. Denote $h = h_{\ell,c}$. By Parseval Identity and the convolution theorem, $\|h * f\|_2^2 = \sum_\alpha \left|\widehat{h}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2$. By definition of h, $\widehat{h}(\alpha) = S_{2^\ell}(\alpha - c)$ for $S_t(\alpha) = \frac{1}{t}\sum_{y=0}^{t-1} \chi_\alpha(y)$ as defined in Proposition 2. The proof follows from the properties guaranteed in Proposition 2; details omitted from this extended abstract. □

We show that when using a good set of queries S condition (*) holds (with high probability over the random noise η).

Lemma 3. *Let $f' = f + \eta$ and all other parameters be as in Theorem 4. Condition (*) holds for f' with probability at least $1 - 1/N^{\Omega(1)}$ over the noise η.*

Proof. Let $S = \bigcup_{\ell=1}^{\log N}(A - B_\ell)$ for $S = S_{N,t,\tau}$ from the good queries $\mathcal{S}$ of Theorem 4. Recall that A is a γ-biased set and the B_ℓ's are $(\gamma, [0..2^\ell])$-biased.

Fix $\ell \in [\lfloor(\log N)\rfloor]$ and $c \in \mathbb{Z}_N$. Denote $B = B_\ell$, $h = h_{\ell,c}$. Observe that $\left|\mathsf{est}_{\ell,c}(f') - \|h * f\|_2^2\right| \le (i) + (ii) + (iii)$ for:

- $(i) := \left|\mathsf{est}_{\ell,c}(f) - \|h * f\|_2^2\right|$
- $(ii) := \left|2\frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{-c}(y)f(x-y)\right)\left(\frac{1}{|B|}\sum_{y\in B}\chi_{-c}(y)\eta(x-y)\right)\right|$
- $(iii) := |\mathsf{est}_{\ell,c}(\eta)|$

We bound each of these terms. By Claim 22, $(i) \le O(\gamma L_1(\widehat{f})^2 \log N)$. By Claims 23-24, with probability at least $1 - 3\exp(-\Omega(|A|\,\tau^2))$, $(ii) + (iii) \le (2 + O(\gamma L_1(\widehat{f})^2 \log N))(2\varepsilon^2 + \varepsilon + O(\tau))$. Thus, for $\gamma = O(\tau/(t^2 \log N))$ and $\varepsilon = O(\tau)$, with probability at least $1 - 3\exp(-\Omega(|A|\,\tau^2))$,

$$\left|\mathsf{est}_{\ell,c}(f') - \|h * f\|_2^2\right| \le O(\tau) \text{ for all } f \text{ s.t. } L_1(\widehat{f}) \le t.$$

By union bound, this holds for *all* $\ell = 1, \dots, \lfloor(\log N)\rfloor$ with probability at least $1 - 3\exp(-\Omega(|A|\,\tau^2))\log N = 1 - 1/N^{\Omega(1)}$ since $|A| \ge \Omega((\ln N)/\tau^2)$ by definition of good sets. □

Claim 22. $(i) \le O(\gamma L_1(\widehat{f})^2 \log N)$.

Proof. Denote $I = [0..2^\ell]$. Define $g_x(y) = \chi_{-c}(y)\overline{f(x-y)}$ for $y \in I$ and $g_x(y) = 0$ otherwise. Then by the definition of $\mathsf{est}_{\ell,c}(f)$ and $\|h * f\|_2^2$,

$$(i) = \left|\mathop{\mathbb{E}}_{x\in A}\left(\mathop{\mathbb{E}}_{y\in B} g_x(y)\right)^2 - \mathop{\mathbb{E}}_{x\in\mathbb{Z}_N}\left(\mathop{\mathbb{E}}_{y\in I} g_x(y)\right)^2\right| \le (i') + (ii') \quad \text{for:}$$

- $(i') := \left|\mathbb{E}_{x\in A}\left(\mathbb{E}_{y\in B}\, g_x(y)\right)^2 - \mathbb{E}_{x\in A}\left(\mathbb{E}_{y\in I}\, g_x(y)\right)^2\right|$
- $(ii') := \left|\mathbb{E}_{x\in A}\left(\mathbb{E}_{y\in I}\, g_x(y)\right)^2 - \mathbb{E}_{x\in\mathbb{Z}_N}\left(\mathbb{E}_{y\in I}\, g_x(y)\right)^2\right|$

We show below that $(i') \le \gamma \cdot L_1(\widehat{f})^2 \cdot O(\log N)$ and $(ii') \le \gamma \cdot L_1(\widehat{f})^2$. Combining these bounds we get that $(i) \le O(\gamma L_1(\widehat{f})^2 \log N)$.

Bounding term (i'). We first get rid of the expectation over $x \in A$ by upper bounding it with its value on a maximizing $x_0 \in A$. We then switch to the Fourier representation of g_{x_0} and rely on B being (γ, I)-biased to bound the difference between the expectations over $y \in B$ and $y \in I$. Finally, we bound the emerging quantity $L_1(\widehat{g_{x_0}})$ (using Proposition 2 and algebraic manipulations). Details omitted from this extended abstract.

Bounding term (ii'). We first observe that the inner expectations are over the same range I and variable. That is, $(ii') = |\mathbb{E}_{x \in A}\, \bar{g}(x) - \mathbb{E}_{x \in \mathbb{Z}_N}\, \bar{g}(x)|$ for $\bar{g}(x) = \left(\mathbb{E}_{y \in I}\, g_x(y)\right)^2$. We then switch to the Fourier representation of $\bar{g}$ and rely on A being γ-biased to bound the difference between the expectations over $x \in A$ and $x \in \mathbb{Z}_N$.

$$(ii') \le \sum_{\alpha \in \mathbb{Z}_N} \left|\widehat{\bar{g}}(\alpha)\right| \left| \mathop{\mathbb{E}}_{x \in A} \chi_\alpha(x) - \mathop{\mathbb{E}}_{x \in \mathbb{Z}_N} \chi_\alpha(x) \right| \le \gamma L_1(\widehat{\bar{g}})$$

Finally we bound the emerging quantity $L_1(\widehat{\bar{g}})$. Observe that $\bar{g} = (h * f)^2$ (since $h * f = \mathbb{E}_{y \in \mathbb{Z}_N} \frac{N}{|I|} \chi_{-c}(y) \overline{f(x-y)} = \mathbb{E}_{y \in I}\, \chi_{-c}(y) \overline{f(x-y)}$). Therefore, $L_1(\widehat{\bar{g}}) \le L_1(\widehat{h * f})^2$ where we use the fact that for any function s, $L_1(\widehat{s^2}) \le L_1(\widehat{s})^2$. Observe further that $L_1(\widehat{h * f})^2 \le L_1(\widehat{f})^2$ because $\left|\widehat{h * f}(\alpha)\right| = \left|\widehat{h}(\alpha)\right| \cdot \left|\widehat{f}(\alpha)\right| \le \left|\widehat{f}(\alpha)\right|$, where the last inequality follows since $\left|\widehat{h}(\alpha)\right| \le 1$ for all α. Combining the above bounds we conclude that $(ii') \le \gamma L_1(\widehat{f})^2$. □

Claim 23. $(ii) \le (1 + O(\gamma L_1(\widehat{f})^2 \log N))(2\varepsilon^2 + \varepsilon + O(\tau))$ *with probability at least* $1 - \exp(-\Omega(|A|\, \tau^2))$.

Proof. By Cauchy-Schwartz inequality, $(ii)^2 \le 4 \cdot (a) \cdot (b)$ for

- $(a) := \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c}(y) f(x-y) \right)^2$
- $(b) := \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c} \eta(x-y) \right)^2$.

To bound (b), observe that $(b) = \mathsf{est}_{\ell,c}(\eta) \le (iii)$. Therefore, by Claim 24, $(b) \le 2\varepsilon^2 + \varepsilon + O(\tau)$ with probability at least $1 - 2\exp(-\Omega(|A|\, \tau^2))$.

To bound (a), observe that $(a) = \mathsf{est}_{\ell,c}(f)$, implying by Claim 22 that $\left|(a) - \|h * f\|_2^2\right| \le O(\gamma L_1(\widehat{f})^2 \log N)$. Next observe that $\|h * f\|_2^2 \le 1$ (since $\|h * f\|_2^2 = \sum_\alpha \left|\widehat{h}(\alpha)\widehat{f}(\alpha)\right|^2$ where $\left|\widehat{h}(\alpha)\right|, \left|\widehat{f}(\alpha)\right| \le 1$ for all α).[3] We conclude therefore that $|(a)| \le 1 + O(\gamma L_1(\widehat{f})^2 \log N)$.

[3] Here, $\left|\widehat{f}(\alpha)\right| \le 1$ because f accepts values in $\mathbb{B}_1$. The bound holds also for unbounded f, provided f is normalized to have $\sum_\alpha \left|\widehat{f}(\alpha)\right|^2 \le 1$.

Combining both bounds we conclude that with probability at least $1 - \exp\left(-\Omega(|A|\,\tau^2)\right)$, $(ii) \le (1 + O(\gamma L_1(\widehat{f})^2 \log N))(2\varepsilon^2 + \varepsilon + O(\tau))$. □

Claim 24. *$(iii) \le 2\varepsilon^2 + \varepsilon + O(\tau)$ with probability at least $1 - 2\exp(-\Omega(|A|\,\tau^2))$.*

Proof. To bound $(iii) = |\mathsf{est}_{\ell,c}(\eta)|$ we rely on the randomness of η. By definition of $\mathsf{est}_{\ell,c}(\eta)$ and the triangle inequality, $|\mathsf{est}_{\ell,c}(\eta)| \le \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}|\eta(x-y)|\right)^2$. Opening the parenthesis, $|\mathsf{est}_{\ell,c}(\eta)| \le (a)+(b)$ for:

– $(a) := \frac{1}{|A|}\sum_{x\in A}\frac{1}{|B|^2}\sum_{y_1\neq y_2\in B}|\eta(x-y_1)|\,|\eta(x-y_2)|$
– $(b) := \frac{1}{|A|}\sum_{x\in A}\frac{1}{|B|^2}\sum_{y\in B}|\eta(x-y)|^2$

Expressions (a) and (b) are averages over the indep. random variables: $v_{x,y_1,y_2} = |\eta(x-y_1)|\,|\eta(x-y_2)|\cdot\frac{|B|}{|B|-1}$ and $v_{x,y} = |\eta(x-y)|^2\cdot\frac{1}{|B|}$ respectively (the factors involving $|B|$ are for proper normalization). We use Chernoff/Hoeffding bound to upper bound expressions (a) and (b) separately, and then apply union bound to upper bound their sum. Details omitted from this extended abstract. □

5 Bit Security Implications

We obtain bit security results as a corollary of our algorithm solving $\mathrm{HNP}^{\mathcal{P},\varepsilon}$.

We set some terminology. Let $G = \{g_p\}$ be a family of generators g_p of $\mathbb{Z}_p^*$. Let $\mathcal{F} = \{f_p\}$ be a family of functions f_p outputting secrets s when given public data $PD_{p,g,s}$ depending on the modulus p, a generator g and the secret s. Think of $\mathcal{F}$ as the underlying hard to compute function. Let $\mathcal{P} = \{P_p\}$ be a family of predicates over $\mathbb{Z}_p^*$. Denote by MB a "magic box" that, given p, g and $PD_{p,g,s}$, outputs $MB(p, g, PD_{p,g,s}) \overset{def}{=} P_p(s)$. We say that:

– $\mathcal{P}$ *is as hard as* $\mathcal{F}$ if there is an algorithm A that, given $PD_{p,g_p,s}$, oracle access to MB, and an advice depending only on p and g_p, outputs the secret s with probability at least $1/poly(\log p)$, while the running time and advice length are polynomial in $\log p$.
– $\mathcal{F}$is $\mathcal{G}$*-accessible* if there is an access algorithm that, given public data $PD_{p,g,s}$ for a secret s, and an element $a \in \mathbb{Z}_{p-1}$, outputs public data $PD_{p,g,s\cdot g^a}$ for the secret $s \cdot g^a \bmod p$.

Theorem 5. *For any $\mathcal{G}$-accessible $\mathcal{F}$ and concentrated $\mathcal{P}$, $\mathcal{P}$ is as hard as $\mathcal{F}$.*

Proof. Fix p and denote $g = g_p$. Let $Adv_{p,g}$ be an advice depending only on p and g as used in Theorem 1 for solving $\mathrm{HNP}^{\mathcal{P},\varepsilon}$ in Algorithm 1. Let $P_{p,s}(a) \overset{def}{=} P_p(s \cdot g^a)$. Observe that given $PD_{p,g,s}$ and oracle access to MB we can simulate oracle access to $P_{p,s}$: For each query a, we compute $PD_{p,g,s\cdot g^a}$ using the access algorithm of $\mathcal{F}$, and output $val = MB(p, PD_{p,g,s\cdot g^a})$. By definition of MB, $val = P_p(s \cdot g^a)$.

The algorithm A runs Algorithm 1 while simulating oracle access to $P_{p,s}$. By Theorem 1, the output is s with probability at least $1/poly(\log p)$. We conclude that $\mathcal{P}$ is as hard as $\mathcal{F}$. $\square$

Let $\mathcal{OK}$ and $\mathcal{EL}'$ denote the underlying hard families of functions in the Okamoto conference key sharing scheme and in the (modified) ElGamal public key encryption scheme as defined in [5]. The analysis of [5] shows that $\mathcal{OK}$ ($\mathcal{EL}'$) is $\mathcal{G}$-accessible. We conclude therefore that for any concentrated predicate $\mathcal{P}$, $\mathcal{P}$ is as hard as computing $\mathcal{OK}$ ($\mathcal{EL}'$). In particular, this holds for $\mathcal{P} = MSB_1$.

Acknowledgments

The author is grateful to Shafi Goldwasser, Igor Shparlinski, Vinod Vaikuntanathan, and Avi Wigderson for helpful discussions.

References

1. Akavia, A.: Learning Noisy Characters, Multiplication Codes and Cryptographic Hardcore Predicates. Ph.D dissertation, defended August 2007, MIT, EECS (February 2008)
2. Akavia, A., Goldwasser, S., Safra, S.: Proving Hard-Core Predicates using List Decoding. In: Proc. of 44th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2003), pp. 146–157. IEEE Computer Society, Los Alamitos (2003)
3. Akavia, A.: Finding significant fourier coefficients deterministically and locally. ECCC Report TR08-102 (2008)
4. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996)
5. Boneh, D., Venkatesan, R.: Rounding in lattices and its cryptographic applications. In: SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms) (1997)
6. Gilbert, A.C., Guha, S., Indyk, P., Muthukrishnan, S., Strauss, M.: Near-optimal sparse fourier representations via sampling. In: Proc. of 34 ACM Annual Symposium on Theory of Computing (STOC 2002), pp. 152–161. ACM Press, New York (2002)
7. Gilbert, A.C., Muthukrishnan, S., Strauss, M.: Improved time bounds for near-optimal sparse fourier representation via sampling. In: Proc. SPIE Wavelets XI (2005)
8. González-Vasco, M.I., Shparlinski, I.: On the security of diffie-hellman bits. In: Proc. Workshop on Cryptography and Computational Number Theory, Singapore, pp. 257–268. Birkhäuser, Basel (2001)
9. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Amer. Stat. Assoc. 58, 13–30 (1963)
10. Iwen, M.A.: A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. CoRR, abs/0708.1211 (2007)
11. Iwen, M.A.: A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In: SODA, pp. 20–29 (2008)

12. Kushilevitz, E., Mansour, Y.: Learning decision trees using the Fourier spectrum. SICOMP 22(6), 1331–1348 (1993)
13. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (1982)
14. Shparlinski, I., Winterhof, A.: A nonuniform algorithm for the hidden number problem in subgroups. In: Bao, F., Deng, R.H., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 416–424. Springer, Heidelberg (2004)
15. Shparlinski, I., Winterhof, A.: A hidden number problem in small subgroups. Math. Comp. 74, 2073–2080 (2005)
16. Terras, A.: Fourier Analysis on Finite Groups and Applications. Cambridge U. Press, Cambridge (1999)

Computational Indistinguishability Amplification: Tight Product Theorems for System Composition[⋆]

Ueli Maurer and Stefano Tessaro

Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
{maurer,tessaros}@inf.ethz.ch

Abstract. Computational indistinguishability amplification is the problem of strengthening cryptographic primitives whose security is defined by bounding the distinguishing advantage of an efficient distinguisher. Examples include pseudorandom generators (PRGs), pseudorandom functions (PRFs), and pseudorandom permutations (PRPs).

The literature on computational indistinguishability amplification consists only of few isolated results. Yao's XOR-lemma implies, by a hybrid argument, that no efficient distinguisher has advantage better than (roughly) $n2^{m-1}\delta^m$ in distinguishing the XOR of m independent n-bit PRG outputs $S_1, \ldots, S_m$ from uniform randomness if no efficient distinguisher has advantage more than δ in distinguishing S_i from a uniform n-bit string. The factor 2^{m-1} allows for security amplification only if $\delta < \frac{1}{2}$: For the case of PRFs, a random-offset XOR-construction of Myers was the first result to achieve *strong* security amplification, i.e., also for $\frac{1}{2} \leq \delta < 1$.

This paper proposes a systematic treatment of computational indistinguishability amplification. We generalize and improve the above product theorem for the XOR of PRGs along five axes. First, we prove the *tight* information-theoretic bound $2^{m-1}\delta^m$ (without factor n) also for the computational setting. Second, we prove results for *interactive* systems (e.g. PRFs or PRPs). Third, we consider the general class of *neutralizing combination constructions*, not just XOR. As an application, this yields the first indistinguishability amplification results for the cascade of PRPs (i.e., block ciphers) converting a weak PRP into an arbitrarily strong PRP, both for single-sided and two-sided queries. Fourth, *strong* security amplification is achieved for a subclass of neutralizing constructions which includes as a special case the construction of Myers. As an application we obtain highly practical optimal security amplification for block ciphers, simply by adding random offsets at the input and output of the cascade. Fifth, we show strong security amplification also for *weakened assumptions* like security against random-input (as opposed to chosen-input) attacks.

A key technique is a generalization of Yao's XOR-lemma to (interactive) systems which is of independent interest.

[⋆] This research was partially supported by the Swiss National Science Foundation (SNF), project no. 200020-113700/1.

1 Introduction

1.1 Security Amplification

The security of all computationally secure cryptographic systems, even of those called "provably secure" in the literature, relies on unproven assumptions about the underlying cryptographic primitives. Typical assumptions are that a certain construction is a one-way function (OWF), a collision-resistant hash function, a pseudorandom generator (PRG), a pseudorandom function (PRF), a pseudorandom permutation (PRP), etc. To weaken these assumptions is both a fundamental challenge in the theory of cryptography and a major goal for the cautious and prudent design of practical cryptographic systems. Many reductions of strong primitives to weak primitives are known. For example, one of the outstanding results is the construction of a PRG from any OWF [13]. However, this reduction, like many other reductions, is highly inefficient and, while of high theoretical value, not of practical relevance.

A specific way to weaken an assumption is to require only that the security property is mildly true. For instance, a δ-OWF can be efficiently inverted with probability at most δ (rather than a negligible quantity for a regular OWF). Similarly, for a δ-PRG no efficient distinguisher has an advantage more than δ in distinguishing its output from a uniform random string. The corresponding definitions of a δ-PRF, a δ-PRP, etc., are straight-forward. Such a weakened assumption is more likely to be true. For example, it is more conservative to only assume that AES is a 0.99-PRP rather than a fully secure PRP.

The natural question is whether several weak primitives can be efficiently combined to obtain a stronger version of the primitive, ideally one with the full-fledged security property.[1] This is called *security amplification*, in some cases hardness amplification. The classical result on security amplification due to Yao [35] is that the parallel composition of m δ-OWFs results in a $(\delta^m + \nu)$-OWF, where ν is some negligible quantity and for any $\delta < 1$, δ^m can be made negligible for large enough m. Security amplifications of a wide range of cryptographic primitives has subsequently been considered, including for example regular OWFs and OWPs [9,11], two-party protocols [1,29,30,34,12], key-agreement and public-key encryption [7,15,16], collision-resistant hash functions [4], and watermarking schemes [17].[2]

The term *indistinguishability amplification* refers to security amplification when the relevant security quantity is the *distinguishing advantage* for the best distinguisher from a certain class of distinguishers, typically the class of efficient distinguishers.

[1] Typically one considers several independent instantiations of the *same* weak primitive, but most results actually hold for several different instantiations.

[2] So-called combiners [14] are another method for relaxing security assumptions: They guarantee that a construction involving several instantiations of a primitive is (fully) secure if at least one (or several, but not all) of them are (fully) secure. However, they do not amplify security of the underlying primitives.

1.2 The XOR-Lemma and Amplification for PRGs

Before we discuss the XOR-lemma, let us compare the prediction advantage and the distinguishing advantage of a biased bit, in an information-theoretic setting, i.e., allowing arbitrary computing power. A bit with bias ϵ takes on the two values with probabilities $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$. When such a bit must be guessed, one would choose the more likely value and be correct with probability $\frac{1}{2} + \epsilon$. To calibrate the guessing advantage, between 0 (when $\epsilon = 0$) and 1 (when the bit is fixed, i.e., $\epsilon = \frac{1}{2}$), one defines the advantage to be 2ϵ. In contrast, the distinguishing advantage is defined as ϵ (with no factor 2) since it is naturally defined for general random variables (not only bits) as the distance of the probability distribution from the uniform one.

As an example, consider two independent bits with biases ϵ_1 and ϵ_2. It is easy to see that the bias of the XOR is $2\epsilon_1\epsilon_2$. For instance, the XOR of a 0.1-biased bit (40/60) and a 0.2-biased bit (30/70) is a 0.04-biased bit (46/54), where $0.04 = 2 \cdot 0.01 \cdot 0.02$. More generally, the bias of the XOR of m bits is 2^{m-1} times the product of the biases. For the XOR of m bit-strings $S_1, \ldots, S_m$ of length n, where S_i has distance δ_i from a uniform n-bit string, the distance from uniform of the XOR of the strings, $S_1 \oplus S_2 \oplus \cdots \oplus S_m$, is bounded by $2^{m-1} \prod_{i=1}^{m} \delta_i$. This bound is tight, as for example the case $n = 1$ discussed above illustrates.

Let us now move to the computational setting, i.e., to Yao's XOR-lemma [35,10], which is much more involved and is another seminal security amplification result. One typically considers a predicate $B(x)$ of the input of a OWF f which is hard to guess when given the output $f(x)$, for uniformly chosen x. But the setting of the XOR-lemma is more general. It states[3] that if for bits $B_1, \ldots, B_m$ the advantage in guessing B_i given some correlated information X_i is at most α_i for any algorithm with complexity t', then no algorithm with complexity t has advantage more than $\prod_{i=1}^{m} \alpha_i + \gamma$ in guessing their XOR-sum, i.e., $B_1 \oplus \cdots \oplus B_m$, given $X_1, \ldots, X_m$, where γ can be made arbitrarily small, at the cost of making t smaller with respect to t'.[4] In terms of distinguishing advantages δ_i, the bound is $2^{m-1} \prod_{i=1}^{m} \delta_i + \gamma$ (for the reasons described above).

Moreover, a standard hybrid argument, to use the unpredictability of bits to prove the indistinguishability of bit-strings, implies an indistinguishability amplification result for PRGs. Consider m independent PRG outputs, $S_1, \ldots, S_m$, each an n-bit string. If no distinguisher with complexity t' has advantage more than δ_i in distinguishing S_i from a uniform random n-bit string, then no distinguisher with complexity (roughly) t has advantage more than $n(2^{m-1} \prod_{i=1}^{m} \delta_i + \gamma)$ in

[3] In fact, one needs a "tight" version of the XOR-lemma for this statement to hold, such as the one by Levin [20,10], or one obtained from a tight hard-core lemma (e.g. [15]) via the techniques of [18].

[4] As usual in complexity-theoretic hardness amplification, we experience an *unavoidable* [31] trade-off between the choice of γ (the tightness of the bound) and the complexity of the reduction.

distinguishing $S_1 \oplus S_2 \oplus \cdots \oplus S_m$ from a uniform random n-bit string.[5] The factor n comes from the hybrid argument over the individual bits of the bit-string.

As explained, the factor 2^{m-1} is unavoidable, since it holds even in the information-theoretic setting. Unfortunately, it means that an amplification can be achieved only if the component constructions are better than $\frac{1}{2}$-secure, i.e., if $\delta_i < \frac{1}{2}$.

1.3 Natural Questions and Previous Results

The above discussion suggests a number of natural questions. (1) Can the factor n in the bound for the XOR of PRGs be eliminated, to obtain a tight bound, namely the equivalent of the information-theoretic counterpart? (2) Can the result be extended to the XOR of PRFs, i.e., primitives for which the security is defined by an *interactive* game, not by the (static) indistinguishability of random variables? (3) If the answer is "yes", can such a result be extended to other constructions, most importantly the cascade of PRPs? (4) Can the factor 2^{m-1} be eliminated so that security amplification from arbitrarily weak components can be achieved? We will answer all these questions positively.

In the information-theoretic setting, questions 2 and 3 were answered by Maurer, Pietrzak, and Renner [24], whose abstract approach we follow, and the special case of permutations had previously been solved by Vaudenay [32,33]. In contrast, there are only a few isolated results on *computational* indistinguishability amplification, which we now discuss. Myers [27] was the first to consider security amplification for PRFs. Interestingly, he did not solve question 2 above, which remained open, but he actually solved part of question 4. More precisely, he showed for the XOR of PRFs, with the modification that for each PRF a random (secret) offset is XORed to the input, that the stronger bound (without the factor 2^{m-1}) can be achieved. However, his treatment is specific for his construction and does not extend to other settings like the cascade of PRPs. Dodis et al. [6] addressed question 2 and gave a positive answer using techniques originating from the setting of hardness amplification of weakly verifiable puzzles [3,19]. However, their focus is on general interactive cryptographic primitives, including for example message authentication codes (MACs), and the resulting bound for the case of PRFs depends on the number of queries the distinguisher is allowed to ask and is not optimal.

Little is known about the cascade of weak PRPs, which is perhaps the case of highest practical interest as it addresses security amplification for block ciphers.[6]

[5] It is not clear to us whether this fact has been published, or is unpublished but well-known folklore, or not so well-known (see also [6] for a similar statement about security amplification for the XOR of PRGs).

[6] Cascades of block ciphers were considered by Even and Goldreich [8] and Maurer and Massey [23], but those results only prove that the cascade is as secure as the strongest component (with no amplification), i.e., that the cascade is a combiner for encryption. Bellare and Rogaway [2] showed a certain security amplification (of a different type) for cascade encryption in the ideal cipher model, which is a purely information-theoretic consideration.

Luby and Rackoff [21] proved an amplification result for the cascade of *two* weak PRPs. This result was extended by Myers [26] to the cascade of a small number of PRPs, but he notes that this result falls short of constructing a (regular) PRP from a weak PRP and states this as an open problem, which we solve.

1.4 Contributions of This Paper

In our attempt at solving the different open questions explained above, we take a very general approach, not targeted at specific constructions. The goal is to develop a deeper and more general understanding and to prove results of a generality that can be useful for other applications.

A first result is a generalization of the XOR-lemma to interactive systems. If a system (as opposed to a random variable for the standard XOR-lemma) of a general type depends on a bit, and no efficient algorithm with access to the system can predict the bit better than with a certain advantage, then the advantage in predicting the XOR of several such bits is bounded by the product of the individual advantages, even if the predictor has complete and arbitrary independent access to all the involved systems.

The XOR of strings or (of the output) of systems, as well as the cascade of systems implementing permutations, are both special cases of a more general concept which was called *neutralizing construction* in [24]. Intuitively, a construction involving several component systems is neutralizing if it is equivalent to an ideal system whenever one component is ideal. For example, the XOR of several PRFs is equivalent to a truly random function if (any) one of the PRFs is replaced by a truly random function.

We prove two tight general product theorems. The first theorem relies on the XOR-lemma and shows that for *all* neutralizing constructions the distinguishing advantage of the combined system is 2^{m-1} times the product of the individual advantages, which is optimal. The second theorem gets rid of the factor 2^{m-1} by considering a special class of *randomized* neutralizing constructions. The applications mentioned in the abstract and the previous sections follow directly from these general theorems.[7] In particular, one application is a highly practical construction for optimal security amplification for block ciphers, simply by adding random offsets at the input and output of the cascade.

1.5 Notational Preliminaries

Throughout this paper, we use calligraphic letters $\mathcal{X}, \mathcal{Y}, \ldots$ to denote sets, upper-case letters $X, Y, \ldots$ to denote random variables, and lower-case letters $x, y, \ldots$ denote the values they take. Moreover, $\mathsf{P}[\mathcal{A}]$ denotes the probability of an event $\mathcal{A}$, while we use the shorthand $\mathsf{P}_X(x) := \mathsf{P}[X = x]$, and denote by P_X the probability distribution of X and by $\mathsf{E}[X]$ its expected value.

[7] For each application of the second theorem, one also needs an information-theoretic indistinguishability proof based on the conditional equivalence of two systems, conditioned on an event that must be proved to be unlikely to occur.

We consider *interactive* randomized stateful algorithms in some a-priori fixed (but otherwise unspecified) RAM model of computation. In particular, such an algorithm keeps a state (consisting say of the memory space it uses), and answers each query depending on the input of this query, some coin flips, the current state (which is possibly updated), and (possibly) one or more queries to an underlying system. It is also convenient to denote by $A[\sigma]$ the algorithm obtained by *setting* the state of A to σ (provided σ is a compatible state), and then behaving according to A's description. Additionally, we say that the algorithm A has *time complexity* t_A (where t_A is a function $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$) if the sum of the length of the description of A and the total number of steps of A is at most $t_A(q, s)$ for all sequences of q queries, all compatible initial states with size s, and all compatible interactions with an underlying system. We use the shorthand $t_A(q) := t_A(q, 0)$.

This paper adopts a *concrete* approach, i.e. we do not use asymptotics and statements are inherently non-uniform. Still, all results can be extended to the uniform setting by using standard techniques. We comment on the necessary changes in the full version of this paper.

2 Discrete Systems and Constructions

Discrete Systems, Constructions, and Distinguishers. This paper deals with the general notion of a (single-interface) *discrete system* $\mathbf{F}$ taking inputs $X_1, X_2, \ldots$ and returning outputs $Y_1, Y_2, \ldots$, where the i-th output Y_i depends (probabilistically) on the first i inputs $X^i = [X_1, \ldots, X_i]$ as well as on all previous $i-1$ outputs $Y^{i-1} = [Y_1, \ldots, Y_{i-1}]$. (If all inputs and outputs are in sets $\mathcal{X}$ and $\mathcal{Y}$, respectively, we call $\mathbf{F}$ an $(\mathcal{X}, \mathcal{Y})$*-system.*) Its input-output behavior is minimally described (see e.g. [22]) by the (infinite) sequence of conditional probability distributions $\mathsf{p}^{\mathbf{F}}_{Y_i|X^iY^{i-1}}$ (for all $i \geq 1$). In general, we use the name "system" (as well as $\mathbf{F}$) interchangeably to denote both the input-output behavior determined by conditional probability distributions and an actual discrete system realizing this behavior. It thus makes sense to say that two systems $\mathbf{F}, \mathbf{G}$ are *equivalent* (denoted $\mathbf{F} \equiv \mathbf{G}$) if they have the same input-output behavior. A *random variable* X is the simplest type of system, which answers each query with the same value X.

With $\mathbf{C}(\cdot)$ we denote a *construction* invoking one or more underlying compatible *subsystems*, whereas $\mathbf{C}(\mathbf{F})$, $\mathbf{C}(\mathbf{F}, \mathbf{G})$, etc denote the systems obtained when $\mathbf{C}$ is instantiated with $\mathbf{F}$ (and $\mathbf{G}$). The shorthand $\mathbf{C}(\mathbf{F}, \cdot)$ indicates the construction that behaves as $\mathbf{C}(\mathbf{F}, \mathbf{G})$ given access to the subsystem $\mathbf{G}$. (All notations extend naturally to constructions with more than two subsystems.) A *distinguisher* $\mathbf{D}$ is a system interacting with another system $\mathbf{F}$ giving inputs $X_1, X_2, \ldots$ and obtaining outputs $Y_1, Y_2, \ldots$, outputting a decision bit after a certain number q of queries depending on the transcript (X^q, Y^q): In particular, we denote as $\mathsf{P}[\mathbf{D}(\mathbf{F}) = 1]$ the probability that it outputs 1.

We say that an interactive algorithm A implements a system $\mathbf{F}$ or a construction $\mathbf{C}(\cdot)$ if it has the same input-output behavior as $\mathbf{F}$ and $\mathbf{C}(\cdot)$, respectively.

In particular, we use A (rather than $\mathbf{F}$) whenever we want to stress that we use the particular implementation A of $\mathbf{F}$.

Distinguishing Advantages. The *distinguishing advantage of a distinguisher* $\mathbf{D}$ *in distinguishing two systems* $\mathbf{F}$ *and* $\mathbf{G}$ is the quantity

$$\Delta^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) := |\mathsf{P}[\mathbf{D}(\mathbf{F}) = 1] - \mathsf{P}[\mathbf{D}(\mathbf{G}) = 1]| .$$

We denote as $\Delta_t(\mathbf{F}, \mathbf{G})$, $\Delta_q(\mathbf{F}, \mathbf{G})$, and $\Delta_{t,q}(\mathbf{F}, \mathbf{G})$ the best distinguishing advantages $\Delta^{\mathbf{D}}(\mathbf{F}, \mathbf{G})$ taken over all distinguishers with time complexity at most t, issuing at most q queries, or both, respectively.

System Composition. Given m systems $\mathbf{F}_1, \ldots, \mathbf{F}_m$, we use the shorthand $\mathbf{F}_1 \| \ldots \| \mathbf{F}_m$ to denote their *parallel composition*, i.e., the system allowing parallel concurrent interaction with the (independent) m systems.[8] Moreover, for $(\mathcal{X}, \mathcal{Y})$-systems $\mathbf{F}$ and $\mathbf{G}$, and a random bit B (with distribution P_B), the system $\langle \mathbf{F}, \mathbf{G} \rangle_B$ acts as $\mathbf{F}$ if $B = 0$, and as $\mathbf{G}$ otherwise. Additionally, for any *quasi-group operation*[9] $\star$ on $\mathcal{Y}$ the $(\mathcal{X}, \mathcal{Y})$-system $\mathbf{F} \star \mathbf{G}$ on input x invokes both $\mathbf{F}, \mathbf{G}$ with input x, obtaining y, y', and returns $y \star y'$.[10] Also, for an $(\mathcal{X}, \mathcal{Y})$-system $\mathbf{P}$ and a $(\mathcal{Y}, \mathcal{Z})$-system $\mathbf{Q}$ we denote with $\mathbf{P} \rhd \mathbf{Q}$ the *cascade of* $\mathbf{P}$ *and* $\mathbf{Q}$, i.e., the system which on input x first invokes $\mathbf{P}$ on this input, and the resulting output is fed into $\mathbf{Q}$ to obtain the final output.

Stateless Systems. We say that a system $\mathbf{F}$ is *stateless* if there exists a conditional probability distribution $\mathsf{p}^{\mathbf{F}}_{Y|X}$ such that $\mathsf{p}^{\mathbf{F}}_{Y_i|X^iY^{i-1}}(y_i, x^i, y^{i-1}) = \mathsf{p}^{\mathbf{F}}_{Y|X}(y_i, x_i)$ for all $y_i, x^i = [x_1, \ldots, x_i]$, and $y^{i-1} = [y_1, \ldots, y_{i-1}]$. Moreover, the system $\mathbf{F}$ is *convex-combination stateless* (*cc-stateless*, for short) if there exists a random variable S and a construction $\mathbf{F}(\cdot)$ (we abuse notation by recycling the letter $\mathbf{F}$) such that $\mathbf{F}(S) \equiv \mathbf{F}$, and $\mathbf{F}(s)$ is stateless for *all* values s taken by S. Depending on the context, S may be e.g. a seed, a key, or an internal function table. A non-trivial example of a cc-stateless system is a randomized encryption scheme, which takes a secret key and encrypts each message with independent randomness. Note that $\langle \mathbf{F}, \mathbf{G} \rangle_B$ is cc-stateless if both $\mathbf{F}, \mathbf{G}$ are cc-stateless.

Random Functions. A *random function* $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$ is an $(\mathcal{X}, \mathcal{Y})$-system which answers consistently, i.e. $X_i = X_j$ implies $Y_i = Y_j$. It is called a *random permutation* if additionally $Y_i = Y_j$ implies $X_i = X_j$. A cc-stateless random function $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$ is in particular such that $\mathbf{F} \equiv \mathbf{F}(S)$ where $\mathbf{F}(\cdot)$ is deterministic and $\mathbf{F}(s)$ is a function $\mathcal{X} \to \mathcal{Y}$ for all s. (This is sometimes called a *keyed function family*, but we also consider the case where s is huge and is hence not a key.) Special cases are a *uniform random function* (URF) $\mathbf{R} : \mathcal{X} \to \mathcal{Y}$ and a *uniform*

[8] The systems do not interact with each other, and each query to the parallel composition is addressed to one of the systems.

[9] That is, given $a, c \in \mathcal{Y}$ (or $b, c \in \mathcal{Y}$) there exists a unique b (a) such that $a \star b = c$. An example is bit-wise XOR $\oplus$ for $\mathcal{Y} = \{0, 1\}^n$, but any group operation is a quasi-group operation as well.

[10] We denote as $\mathbf{F}_1 \star \cdots \star \mathbf{F}_m$ the system $(\cdots((\mathbf{F}_1 \star \mathbf{F}_2) \star \mathbf{F}_3) \cdots) \star \mathbf{F}_m$.

random permutation (URP) $\mathbf{P} : \mathcal{X} \to \mathcal{X}$ that realize a uniformly chosen function $\mathcal{X} \to \mathcal{Y}$ and permutation $\mathcal{X} \to \mathcal{X}$, respectively. We denote as $\mathbf{F}(s, x)$ the evaluation of $\mathbf{F}$ with key s and input x.

Informally, in an asymptotic setting, it is convenient to say that an efficient $\mathbf{F}(\cdot)$ is a *δ-pseudorandom function* (PRF) if $\Delta_{t,q}(\mathbf{F}(S), \mathbf{R}) \leq \delta + \mathsf{negl}$ for a (short) key S, a URF $\mathbf{R}$, all polynomial t and q, and some negligible[11] function negl. Analogously, if an efficient $\mathbf{Q}(\cdot)$ implements a permutation for all keys, it is called a *δ-pseudorandom permutation (PRP)* if $\Delta_{t,q}(\mathbf{Q}(S), \mathbf{P}) \leq \delta + \mathsf{negl}$ for a URP $\mathbf{P}$ and for all polynomial t and q.

The inverse $\mathbf{Q}^{-1}$ of a cc-stateless random permutation $\mathbf{Q}$ is well-defined, and $\langle \mathbf{Q} \rangle$ is the system accepting *forward queries* $(x, +)$ (answered by $\mathbf{Q}(s, x)$ on key s) and *backward queries* $(y, -)$ (answered as $\mathbf{Q}^{-1}(s, y)$). In particular $\langle \mathbf{Q} \rangle \triangleright \langle \mathbf{Q}' \rangle$ stands for the system $\langle \mathbf{Q} \triangleright \mathbf{Q}' \rangle$. An efficient $\mathbf{Q}(\cdot)$ is called a *δ-two-sided PRP*[12] if $\Delta_{t,q}(\langle \mathbf{Q}(S) \rangle, \langle \mathbf{P} \rangle) \leq \epsilon + \mathsf{negl}$ for all polynomial q and t. (Of course, one assumes that backward queries can be computed efficiently given s.)

Neutralizing Constructions. A construction $\mathbf{C}(\cdot)$ is *neutralizing* [24] for systems $\mathbf{F}_1, \ldots, \mathbf{F}_m$ and ideal systems $\mathbf{I}_1, \ldots, \mathbf{I}_m$, if for $\mathbf{S}_i \in \{\mathbf{F}_i, \mathbf{I}_i\}$ $(i = 1, \ldots, m)$ we have $\mathbf{C}(\mathbf{S}_1, \ldots, \mathbf{S}_m) \equiv \mathbf{C}(\mathbf{I}_1, \ldots, \mathbf{I}_m)$ whenever there exists some i with $\mathbf{S}_i = \mathbf{I}_i$.[13]

Every quasi-group operation $\star$ on a set $\mathcal{Y}$ induces a construction $\mathbf{C}(\cdot)$ such that $\mathbf{C}(\mathbf{F}, \mathbf{G}) := \mathbf{F} \star \mathbf{G}$ which is neutralizing for random functions $\mathbf{F}, \mathbf{G} : \mathcal{X} \to \mathcal{Y}$ and ideal systems $\mathbf{I}, \mathbf{J}$ being independent URFs. In particular, $\mathbf{I} \star \mathbf{J}$ is also a URF. As a special case, this result holds for random variables X, Y over $\mathcal{Y}$, the ideal systems being uniform random elements of $\mathcal{Y}$. Moreover, the cascade operator $\triangleright$ induces a construction $\mathbf{C}'(\cdot)$ with $\mathbf{C}'(\mathbf{Q}_1, \mathbf{Q}_2) := \mathbf{Q}_1 \triangleright \mathbf{Q}_2$ which is neutralizing for any two cc-stateless random permutations $\mathbf{Q}_1, \mathbf{Q}_2 : \mathcal{X} \to \mathcal{X}$ (in fact $\mathbf{Q}_1$ can possibly be stateful) with ideal systems $\mathbf{I}, \mathbf{J}$ both URPs $\mathcal{X} \to \mathcal{X}$. In particular, $\mathbf{I} \triangleright \mathbf{J}$ is also a URP. If $\mathbf{Q}_1$ is cc-stateless, then the same result holds even in the two-sided case for $\langle \mathbf{Q}_1 \rangle$ and $\langle \mathbf{Q}_2 \rangle$ (with ideal system $\langle \mathbf{P} \rangle$ for a URP $\mathbf{P}$). Both constructions extend naturally to an arbitrary number of subsystems.

3 A General Product Theorem for Neutralizing Constructions

This section presents a very general product theorem showing computational indistinguishability for *every* neutralizing construction. This result relies on a

[11] Recall that a function $\nu : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible* if it vanishes faster than the inverse of any polynomial.

[12] In the literature the name *strong* PRP is commonly used, but this term is slightly confusing in the context of this paper.

[13] Neutralizing constructions capture the notion of a *combiner* [14] for computational indistinguishability properties: Whenever at least one system $\mathbf{S}_i$ is computationally indistinguishable from $\mathbf{I}_i$, then $\mathbf{C}(\mathbf{S}_1, \ldots, \mathbf{S}_m)$ is computationally indistinguishable from $\mathbf{C}(\mathbf{I}_1, \ldots, \mathbf{I}_m)$.

generalization of Yao's XOR-Lemma to discrete interactive systems, which is presented first, and is of independent interest.

3.1 The Generalized XOR-Lemma

System-Bit Pairs. A *system-bit pair* is a system of the form $(\mathbf{F}, B)$, where $B \in \{0,1\}$ is a bit value, which is (generally) correlated with the system $\mathbf{F}$. This can formally be described by the distribution P_B of B and the two systems $\mathbf{F}_0$ and $\mathbf{F}_1$ conditioned on the value taken by B, i.e. $(\mathbf{F}, B) = (\langle \mathbf{F}_0, \mathbf{F}_1 \rangle_B, B)$. A possible system-bit pair is a URF $\mathbf{R} : \{0,1\}^m \to \{0,1\}$ and the parity of its function table. The following quantity characterizes the performance of an adversary[14] $\mathbf{A}$ in guessing the bit B when given access to $\mathbf{F}$ only.

Definition 1. *The* guessing advantage of an adversary $\mathbf{A}$ in guessing B *for a system-bit pair* $(\mathbf{F}, B)$ *is the quantity* $\Gamma^{\mathbf{A}}(\mathbf{F}, B) := 2 \cdot \mathsf{P}[\mathbf{A}(\mathbf{F}) = B] - 1$. *Additionally, we denote as* $\Gamma_{t,q}(\mathbf{F}, B)$ *the maximal guessing advantage* $\Gamma^{\mathbf{A}}(\mathbf{F}, B)$ *taken over all q-query adversaries* $\mathbf{A}$ *with complexity at most t.*

Note that $\Gamma^{\mathbf{A}}(\mathbf{F}, B) \in [-1, 1]$, where 1 means that $\mathbf{A}$ is able to perfectly predict B by interacting with $\mathbf{F}$, while -1 means that $\mathbf{A}$ is never correct.[15] The following connection between the guessing and the distinguishing advantages is well known (cf. e.g. [24]).

Lemma 1. *For all* $\mathbf{F}$, $\mathbf{G}$, *and* $\mathbf{D}$, *we have* $\Delta^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) = \left|\Gamma^{\mathbf{D}}(\langle \mathbf{F}, \mathbf{G} \rangle_B, B)\right|$ *for a uniform bit* $B \in \{0,1\}$.

The XOR-Lemma. Given m system-bit pairs $(\mathbf{G}_1, B_1), \ldots, (\mathbf{G}_m, B_m)$, we are interested in the advantage $\Gamma_{t,q_1,\ldots,q_m}(\mathbf{G}_1 \| \cdots \| \mathbf{G}_m, B_1 \oplus \cdots \oplus B_m)$ of guessing the bit $B_1 \oplus \cdots \oplus B_m$ given *parallel* access to the systems $\mathbf{G}_1, \ldots, \mathbf{G}_m$, where at most q_i queries to each system $\mathbf{G}_i$ are allowed. That is, we consider the most general attack where the adversary can query each subsystem $\mathbf{G}_i$ individually at most q_i times, *adaptively* depending on the answers of queries to other subsystems. We show that the advantage is upper bounded by the *product* of the individual advantages $\Gamma_{t',q'}(\mathbf{G}_i, B_i)$ for $i = 1, \ldots, m$ (for appropriate t', q'), with an extra additive term $\gamma > 0$ which can be made arbitrarily small (but influences the efficiency of the reduction). The result holds provided that all but one of the system-bit pairs are cc-stateless. (Note that the fact that $(\mathbf{G}_i, B_i)$ is cc-stateless implies that $\mathbf{G}_i$ is cc-stateless, but the converse is not always true.) Our result generalizes the original XOR-lemma by Yao [35,10], which considered the special case of system-bit pairs (X_i, B_i), where X_i is a random variable.

We stress that our result only requires the ability to efficiently implement the cc-stateless system-bit pairs $(\mathbf{G}_i, B_i) = (\mathbf{G}_i(S), B_i(S))$. This may be possible, for instance by using a *stateful* algorithm, even if $\mathbf{G}(\cdot)$ and $B(\cdot)$ themselves are

[14] We stress that distinguishers and adversaries are objects of the same type. The name adversary is used to stress the fact that we are not exclusively considering a distinguishing scenario.

[15] In particular, flipping the output bit of such a $\mathbf{A}$ yields one which is always correct.

not efficiently computable: In fact, S may even be exponentially large. As an example, the aforementioned system-bit pair $(\mathbf{R}, B)$, where $\mathbf{R} : \{0,1\}^n \to \{0,1\}$ is a URF, and B is the parity of its function table, is clearly cc-stateless, and can efficiently be implemented by first sampling a random B, and then answering queries to $\mathbf{R}$ with independent random bits, with the exception of the last one, which is answered so that the parity equals B.

In the following, we define the quantity $\varphi := 2\left(\frac{24m}{\gamma}\right)^2 \cdot \ln\left(\frac{7m}{\gamma}\right)$ for understood m and γ. Also, t_{G_i} and s_{G_i} are the time and space[16] complexities of some implementation G_i of the system $\mathbf{G}_i$, whereas $t_{(G_i,B_i)}$ is the time-complexity of an implementation of the pair $(\mathbf{G}_i, B_i)$. (Note that an efficient implementation of the latter implies one for the former, but we allow for this distinction.) For all i, we denote $l_i := s_{G_i}(q_i \cdot \varphi)$ and $l_{<i} := \sum_{j=1}^{i-1} l_j$ (for understood $q_1, \ldots, q_{m-1}$).

Theorem 1 (XOR-Lemma). *Let $(\mathbf{G}_1, B_1), \ldots, (\mathbf{G}_{m-1}, B_{m-1})$ be cc-stateless system-bit pairs, and let $(\mathbf{G}_m, B_m)$ be an arbitrary system-bit pair. For all t, $q_1, \ldots, q_m$, $\gamma > 0$,*

$$\Gamma_{t,q_1,\ldots,q_m}(\mathbf{G}_1 \| \ldots \| \mathbf{G}_m, B_1 \oplus \cdots \oplus B_m) \leq \prod_{i=1}^{m} \Gamma_{t'_i,q'_i}(\mathbf{G}_i, B_i) + \gamma,$$

where $t'_i := l_{<i} + \varphi \cdot \left[t + \mathcal{O}\left(\sum_{j=1}^{i-1} t_{G_j}(q_j, l_j) + \sum_{j=i+1}^{m} t_{(G_j,B_j)}(q_j)\right)\right]$ and $q'_i := \varphi \cdot q_i$ for $i = 1, \ldots, m-1$, whereas $t_m := l_{<m} + t + \mathcal{O}\left(\sum_{j=1}^{m-1} t_{G_j}(q_j, l_j)\right)$ and $q'_m := q_m$.

The asymmetry of our proof technique allows $(\mathbf{G}_m, B_m)$ to be fully stateful.[17] Furthermore, both t'_m and q'_m are much smaller then the corresponding terms t'_i and q'_i for $i = 1, \ldots, m-1$. The following paragraph provides a proof sketch for the case $m = 2$. The full proof is deferred to the full version of this paper.

Proof Idea for $m = 2$. The proof follows similar lines as Levin's proof of the XOR-lemma [20,10], but with some major differences due to the peculiarities of reactive systems. For simplicity, we let $(\mathbf{F}, B) = (\mathbf{G}_1, B_1)$ and $(\mathbf{G}, C) = (\mathbf{G}_2, B_2)$. Let $\mathbf{A}$ be an adversary with $\Gamma^{\mathbf{A}}(\mathbf{F} \| \mathbf{G}, B \oplus C) > \delta \cdot \epsilon + \gamma$. We show that either there exists an adversary $\mathbf{A}'$ such that $\Gamma^{\mathbf{A}'}(\mathbf{F}, B) > \delta$ or there exists an adversary $\mathbf{A}''$ such that $\Gamma^{\mathbf{A}''}(\mathbf{G}, C) > \epsilon$, contradicting the assumed hardness of $(\mathbf{F}, B)$ and/or $(\mathbf{G}, C)$. The time complexities of $\mathbf{A}'$ and $\mathbf{A}''$ are strictly related to the one of $\mathbf{A}$. Recall that the pair $(\mathbf{F}, B) = (\mathbf{F}(S), B(S))$ is cc-stateless, and for all values s taken by the random variable S we define

$$\alpha_1(s) := \Gamma^{\mathbf{A}}(\mathbf{F}(s) \| \mathbf{G}, 1 \oplus C) \text{ and } \alpha(s) := \Gamma^{\mathbf{A}}(\mathbf{F}(s) \| \mathbf{G}, B(s) \oplus C).$$

[16] i.e. the maximal size of the state after the given number of queries.

[17] An orthogonal generalization of the XOR-lemma for stateful interactive systems was proposed by Halevi and Rabin [12]. However, it relies on *sequential* (rather than parallel) access to the systems $\mathbf{G}_1, \ldots, \mathbf{G}_m$, which is not sufficient for the applications of this paper.

By definition, $\mathsf{E}[\alpha(S)] > \delta \cdot \epsilon + \gamma$. Moreover $\alpha(s) = \alpha_1(s)$ if $B(s) = 1$, and $\alpha(s) = -\alpha_1(s)$ otherwise. This implies that $\alpha_1(S)$ has good correlation with $B(S)$, as an adversary $\mathbf{A}'$ outputting 1 with probability $\frac{1}{2} + \frac{\alpha_1(s)}{2}$ (when given access to $\mathbf{F}(s)$) has advantage at least $\delta \cdot \epsilon + \gamma$. If $|\alpha_1(s)| = |\alpha(s)| \leq \epsilon$ holds for all s, then the advantage can be amplified to be larger than δ by outputting 1 with probability $\frac{1}{2} + \frac{\alpha_1(s)}{2\epsilon}$. Of course, $\mathbf{A}'$ does not know $\alpha_1(s)$, but a statistical estimate can be obtained by repeated interaction with $\mathbf{F}(s)$, as it is stateless: The term γ compensates the possible estimation error.

Note that the existence of a single value s with the property that $|\alpha_1(s)| > \epsilon$ implies that there exists a bit b such that the adversary $\mathbf{A}'' := \mathbf{A}(\mathbf{F}(s) \| \cdot) \oplus b$ has advantage larger than ϵ in guessing C from $\mathbf{G}$, i.e., $\mathbf{A}''$ is the adversary that simulates the execution of $\mathbf{A}$ with the parallel composition of $\mathbf{F}(s)$ and the given system $\mathbf{G}$, and outputs $\mathbf{A}$'s output XORed with b. But such adversary $\mathbf{A}''$ is not necessarily efficient, because an efficient implementation of $\mathbf{F}(s)$ may not exist. To overcome this issue, we show that for the above adversary $\mathbf{A}'$ to succeed, it is sufficient that the probability over the choice of S that $|\alpha_1(S)| > \epsilon + \gamma/4$ is smaller than $\gamma/4$. Furthermore, if this probability is at least $\gamma/4$, a probabilistic argument yields a (sufficiently) small state σ for the (efficient) implementation F of $\mathbf{F}$ and a (fixed) bit b such that the *efficient* adversary $\mathbf{A}'' := \mathbf{A}(F[\sigma] \| \cdot) \oplus b$ achieves advantage at least ϵ.

3.2 A Product Theorem from the XOR-Lemma

Throughout this section, let $\mathbf{C}(\cdot)$ be a neutralizing construction for systems $\mathbf{F}_1, \ldots, \mathbf{F}_m$, $\mathbf{I}_1, \ldots, \mathbf{I}_m$ (of which all but $\mathbf{F}_m$ and $\mathbf{I}_m$ have to be cc-stateless). We provide a very general product theorem upper bounding the distinguishing advantage $\Delta_{t,q}(\mathbf{C}(\mathbf{F}_1, \ldots, \mathbf{F}_m), \mathbf{C}(\mathbf{I}_1, \ldots, \mathbf{I}_m))$ in terms of the individual advantages $\Delta_{t'_i,q'_i}(\mathbf{F}_i, \mathbf{I}_i)$ (for some related t'_i, q'_i). The theorem is a computational version of the information-theoretic product theorem from [24]: In particular, we inherit the same bounds, with an unavoidable additive term.

The theorem relies on the canonical implementation $\langle F_i, I_i \rangle_{B_i}$ of $\langle \mathbf{F}_i, \mathbf{I}_i \rangle_{B_i}$ which chooses a random bit $B_i \in \{0, 1\}$ and answers each query using the implementations F_i and I_i (with respective complexities t_{F_i} and t_{I_i}) of $\mathbf{F}_i$ or of $\mathbf{I}_i$, respectively, depending on the value of B_i. (B_i is in particular part of the state.) It can be implemented with complexity $t_{\langle F_i, B_i \rangle_{B_i}}(q, s) = \max\{t_{F_i}(q, s), t_{I_i}(q, s)\} + \mathcal{O}(1)$. This also yields an implementation of $(\langle \mathbf{F}_i, \mathbf{I}_i \rangle_{B_i}, B_i)$ with the same complexity (by additionally outputting the bit B_i). Finally, we let l_i and $l_{<i}$ as above be defined with respect to $\langle F_i, I_i \rangle_{B_i}$, and let t_C be the time complexity of an efficient implementation of $\mathbf{C}(\cdot)$.

Theorem 2 (Product Theorem). *Let $\mathbf{C}(\cdot)$ be as above, and let $q > 0$ be such that $\mathbf{C}(\cdot)$ makes q_i queries to its i-th subsystem when invoked q times. Then, for all $t, \gamma > 0$, if $\Delta_{t'_i,q'_i}(\mathbf{F}_i, \mathbf{I}_i) \leq \frac{1}{2}$ for all $i = 1, \ldots, m-1$,*

$$\Delta_{t,q}(\mathbf{C}(\mathbf{F}_1, \ldots, \mathbf{F}_m), \mathbf{C}(\mathbf{I}_1, \ldots, \mathbf{I}_m)) \leq 2^{m-1} \cdot \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\mathbf{F}_i, \mathbf{I}_i) + 2\gamma,$$

where $t'_i := l_{<i} + \varphi \cdot \left[t + t_C(q) + \mathcal{O}\left(\sum_{j=1}^{i-1} t_{\langle F_j, I_j\rangle_{B_j}}(q_j, l_j) + \sum_{j=i+1}^{m} t_{\langle F_j, I_j\rangle_{B_j}}(q_j)\right)\right]$ *and* $q'_i := \varphi \cdot q_i$ *for all* $i = 1, \ldots, m-1$, *whereas* $t'_m := l_{<m} + t + t_C(q) + \mathcal{O}\left(\sum_{j=1}^{m-1} t_{\langle F_j, I_j\rangle_{B_j}}(q_j, l_j)\right)$ *and* $q'_m := q_m$.

PROOF SKETCH. We present a proof sketch of the above theorem for the case $m = 2$. For simplicity, let $\mathbf{F}_1 = \mathbf{F}$, $\mathbf{F}_2 = \mathbf{G}$, $\mathbf{I}_1 = \mathbf{I}$, and $\mathbf{I}_2 = \mathbf{J}$. The core of the proof is a generic argument (i.e. it holds for all distinguishers, regardless of their computing power) reducing the task of upper bounding the distinguishing advantage for a neutralizing construction to the setting of the XOR-lemma.[18] It is easy to verify that (also cf. [24])

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})) &= 2 \cdot \Delta^{\mathbf{D}}(\langle \mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_B, \mathbf{C}(\mathbf{I}, \mathbf{J})) \\ &= \left|\Gamma^{\mathbf{D}}(\langle\langle \mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_B, \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_{B'}, B')\right|,\end{aligned}$$

where B and B' are independent uniformly distributed random bits. Note that conditioned on $B' = 0$, the system $\langle\langle \mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_B, \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_{B'}$ behaves as $\mathbf{C}(\mathbf{F}, \mathbf{G})$ with probability $\frac{1}{2}$, and as $\mathbf{C}(\mathbf{I}, \mathbf{J})$ otherwise. On the other hand, conditioned on $B' = 1$ it always behaves as $\mathbf{C}(\mathbf{I}, \mathbf{J})$. In particular, this implies that (for independent uniform random bits B_1, B_2)

$$\left(\langle\langle \mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_B, \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_{B'}, B'\right) \equiv \left(\mathbf{C}(\langle \mathbf{F}, \mathbf{I}\rangle_{B_1}, \langle \mathbf{G}, \mathbf{J}\rangle_{B_2}), B_1 \oplus B_2\right),$$

because of the neutralizing property. We thus obtain

$$\Gamma^{\mathbf{D}}(\langle\langle \mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_B, \mathbf{C}(\mathbf{I}, \mathbf{J})\rangle_{B'}, B') = \Gamma^{\mathbf{D}}(\mathbf{C}(\langle \mathbf{F}, \mathbf{I}\rangle_{B_1}, \langle \mathbf{G}, \mathbf{J}\rangle_{B_2}), B_1 \oplus B_2)$$

and we conclude the proof by "absorbing" the computation of $\mathbf{C}(\cdot)$ into $\mathbf{D}$, clearly without modifying the advantage. Using the XOR-lemma (Theorem 1) for $m = 2$ we obtain

$$\begin{aligned}\Delta_{t,q}(\mathbf{C}(\mathbf{F}, \mathbf{G}), \mathbf{C}(\mathbf{I}, \mathbf{J})) &\leq 2 \cdot \Gamma_{t+t_C(q), q_1, q_2}(\langle \mathbf{F}, \mathbf{I}\rangle_{B_1}, \langle \mathbf{G}, \mathbf{J}\rangle_{B_2}, B_1 \oplus B_2) \\ &\leq 2 \cdot \Gamma_{t'_1, q'_1}(\langle \mathbf{F}, \mathbf{I}\rangle_{B_1}, B_1) \cdot \Gamma_{t'_2, q'_2}(\langle \mathbf{G}, \mathbf{J}\rangle_{B_2}, B_2) + 2\gamma.\end{aligned}$$

for appropriate t'_1, q'_1 and t'_2, q'_2. Extending the proof to *arbitrary* neutralizing constructions for $m > 2$ requires some extra care. The details can be found in the full version of this paper.

3.3 Applications of Theorem 2

SUMS OF PRFS. Let $\mathbf{F}_1, \ldots, \mathbf{F}_m : \mathcal{X} \to \mathcal{Y}$ be cc-stateless random functions (in fact, $\mathbf{F}_m$ can possibly be stateful), and let $\star$ be a quasi-group operation on $\mathcal{Y}$. The operator $\star$ is neutralizing, as discussed in Section 2, for $\mathbf{F}_1, \ldots, \mathbf{F}_m$ and ideal systems $\mathbf{I}_1 = \cdots = \mathbf{I}_m = \mathbf{R}$, where $\mathbf{R} : \mathcal{X} \to \mathcal{Y}$ is a URF. In order to simplify the time complexity statements, we assume that there exist efficient algorithms

[18] A similar argument was implicitly used in the information-theoretic product theorem of [24].

implementing $\mathbf{F}_i(\cdot)$ such that $\mathbf{F}_i(s,x)$ is computed in time t_{F_i} given s and x (this holds in the interesting case where we apply the result to PRFs) and elements of $\mathcal{Y}$ can be encoded using $\ell \approx \log|\mathcal{Y}|$ bits. Note that the canonical implementation of $\mathbf{R}$ keeps a linearly-growing state of size $s = \mathcal{O}(q \cdot \ell)$ after q queries, and answers each query in time $\mathcal{O}(\log(s))$. Therefore, with $t_{\langle F_i,R\rangle_{B_i}}(q,s) = \mathcal{O}(q \cdot \max\{t_{F_i}, \log(s+q\ell)\})$ and $l_{<i} = \mathcal{O}((i-1)\varphi q\ell)$, we apply Theorem 2 to obtain the following result (we tacitly assume that all advantages are bounded by $\frac{1}{2}$):

Corollary 1. *For all $t,q,\gamma > 0$,*

$$\Delta_{t,q}(\mathbf{F}_1 \star \cdots \star \mathbf{F}_m, \mathbf{R}) \leq 2^{m-1} \cdot \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\mathbf{F}_i, \mathbf{R}) + 2\gamma.$$

A version of this result with weaker bounds was shown by Dodis et al. [6] for $\star = \oplus$. (Their bounds depend in particular on the number of queries.) We remark that the analogous result for PRGs follows as a special case, since a PRG can be seen as a one-input PRF.

In the asymptotic setting, if $\mathbf{F}(\cdot)$ is a δ-PRF (for some $\delta < \frac{1}{2}$), it follows that $\mathbf{F}(S_1)\star\cdots\star\mathbf{F}(S_m)$, for independent keys $S_1,\ldots,S_m$, is a $2^{m-1}\cdot\delta^m$-PRF: For t,q polynomial (in n), we have $\Delta_{t,q}(\mathbf{F}(S_1)\star\cdots\star\mathbf{F}(S_m),\mathbf{R}) \leq 2^{m-1}\cdot\delta^m + \mathsf{negl} + 1/p(n)$ for all polynomials p, as both t'_i and q'_i are polynomial as well.

Cascade of PRPs. Let $\mathbf{P} : \{0,1\}^n \to \{0,1\}^n$ be a URP and let $\mathbf{Q}_1,\ldots,\mathbf{Q}_m : \{0,1\}^n \to \{0,1\}^n$ be cc-stateless random permutations. Recall that the $\rhd$ operator is neutralizing for $\mathbf{Q}_1,\ldots,\mathbf{Q}_m$ (all with ideal system $\mathbf{P}$), as well as for $\langle\mathbf{Q}_1\rangle,\ldots,\langle\mathbf{Q}_m\rangle$ (all with ideal system $\langle\mathbf{P}\rangle$). As above, we assume that both $\mathbf{Q}_i(s,x)$ and $\mathbf{Q}_i^{-1}(s,y)$ are computable in time t_{Q_i}. Furthermore, simulating the URP $\mathbf{P}$ (as well as the two-sided URP $\langle\mathbf{P}\rangle$) requires the same complexity as implementing a URF. Therefore, with $t_{\langle Q_i,P\rangle_{B_i}}(q,s) = t_{\langle\langle Q_i\rangle,\langle P\rangle\rangle_{B_i}}(q,s) = \mathcal{O}(q\cdot\max\{t_{Q_i},\log(s+qn)\})$ and $l_{<i} = \mathcal{O}((i-1)\varphi qn)$, Theorem 2 yields the following corollary:

Corollary 2. *For all $t,q,\gamma > 0$,*

$$\Delta_{t,q}(\mathbf{Q}_1 \rhd \cdots \rhd \mathbf{Q}_m, \mathbf{P}) \leq 2^{m-1} \cdot \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\mathbf{Q}_i, \mathbf{P}) + 2\gamma,$$

and

$$\Delta_{t,q}(\langle\mathbf{Q}_1\rangle \rhd \cdots \rhd \langle\mathbf{Q}_m\rangle, \langle\mathbf{P}\rangle) \leq 2^{m-1} \cdot \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\langle\mathbf{Q}_i\rangle, \langle\mathbf{P}\rangle) + 2\gamma.$$

We remark that this is the first result considering *two-sided* PRPs, and even in the one-sided setting only the case $m = 2$ was considered by Luby and Rackoff [21], and subsequently extended to $m = \mathcal{O}(\log n)$ by Myers [26].

Furthermore, we note that $\mathbf{Q}_1$ is allowed to be stateful in the one-sided case, as Theorem 2 allows one system to be stateful: In fact, $\rhd$ is not necessarily neutralizing whenever at least two permutations are stateful.

4 A Strong Product Theorem for Randomized Neutralizing Constructions

4.1 A Product Theorem from Self-independence

Since Theorem 2 holds for arbitrary neutralizing constructions, one cannot avoid the factor 2^{m-1} in the bound. This section shows that a subclass of neutralizing constructions satisfying a simple information-theoretic property yield a *strong* product theorem, i.e., the obtained upper bound is roughly the product of the individual advantages.

Self-Independence. The notion of self-independence of an ideal system $\mathbf{I}$ under a construction $\mathbf{C}(\cdot)$ captures the fact that a computationally unbounded distinguisher cannot tell apart the scenario where the *same* instance of $\mathbf{I}$ is accessed through independent instances of $\mathbf{C}(\cdot)$ from the setting where each instance of $\mathbf{C}(\cdot)$ accesses an independent instance of $\mathbf{I}$.

Definition 2. *The system* $\mathbf{I}$ *is* η-self-independent under $\mathbf{C}(\cdot)$ *for a function* $\eta : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{\geq 0}$*, if for all* $q, \lambda > 0$*, the best (information-theoretic) distinguishing advantage when allowing q queries to each subsystem satisfies*

$$\Delta_{q,\ldots,q}(\mathbf{C}_1(\mathbf{I}) \| \ldots \| \mathbf{C}_\lambda(\mathbf{I}), \mathbf{C}_1(\mathbf{I}_1) \| \ldots \| \mathbf{C}_\lambda(\mathbf{I}_\lambda)) \leq \eta(q, \lambda),$$

where $\mathbf{C}_1(\cdot), \ldots, \mathbf{C}_\lambda(\cdot)$ *and* $\mathbf{I}_1, \ldots, \mathbf{I}_\lambda$ *are independent copies of* $\mathbf{C}(\cdot)$ *and* $\mathbf{I}$*, respectively.*

As an example, consider the construction $\mathbf{C}(\cdot)$ which generates a (secret) random n-bit offset Z, and given access to a random function $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$, $\mathbf{C}(\mathbf{F})$ returns $\mathbf{F}(x \oplus Z)$ upon each query x. It is not hard to show, e.g. using the tools from [22], that a URF $\mathbf{R} : \{0,1\}^n \to \{0,1\}^n$ is η-self-independent under $\mathbf{C}(\cdot)$ for $\eta(q, \lambda) \leq \frac{q^2\lambda^2}{2} \cdot 2^{-n}$, i.e., the probability that for some distinct $i \neq j$ the instances $\mathbf{C}_i(\cdot)$ and $\mathbf{C}_j(\cdot)$ invoke $\mathbf{R}$ with the same input.

Restricted Attacks on Cryptographic Functions. Indistinguishability-based security definitions can also be weakened by restricting the distinguisher's access to the given system. For instance, the standard PRF notion considering an (adaptive) *chosen-input attack* can be weakened to non-adaptive chosen-input attacks or even *(known) random-input* attacks. (Keyed functions which are secure under the latter notion are usually called *weak PRFs* [28] in the literature.[19]) This is conveniently modeled by letting the distinguisher access either of $\mathbf{E}(\mathbf{F})$ and $\mathbf{E}(\mathbf{G})$, where the construction $\mathbf{E}(\cdot)$ enforces a particular type of access, and $\mathbf{F}$ and $\mathbf{G}$ are the systems to be distinguished. For a chosen-input attack, $\mathbf{E}$ would just give full access to the underlying system (i.e. $\mathbf{E}(\cdot)$ is the *identity*), and the following are two additional examples:

[19] The name is slightly misleading within the context of this paper, as it can been used [27] to describe an ϵ-PRF for a non-negligible $\epsilon < 1$.

– *Random-input attacks* against an $(\mathcal{X}, \mathcal{Y})$-system are modeled by $\mathbf{K}(\cdot)$ that, upon each invocation (with some dummy input), generates a fresh uniformly-chosen element $r \in \mathcal{X}$, makes a query with input r to the given subsystem, obtaining $y \in \mathcal{Y}$, and returns (r, y).
– For a quasi-group operation $*$ on $\mathcal{X}$ (usually $\oplus$), a *random-offset attack* is modeled by a construction $\mathbf{Z}(\cdot)$ which initially generates a random offset $Z \in \mathcal{X}$, and upon each invocation with input $x \in \mathcal{X}$, makes a query to the given subsystem with input $x \star Z$, and outputs the returned value y. (To our knowledge, this notion was not previously considered in the literature.)

A feature of the product theorem of this section is that it is easily applicable also to the restricted-access case.

The Product Theorem. In the following, let $\mathbf{C}(\cdot)$ be a neutralizing construction for systems $\mathbf{F}_1, \ldots, \mathbf{F}_m$ and ideal system $\mathbf{I}_1, \ldots, \mathbf{I}_m$, all of which (with the possible exception of $\mathbf{F}_m$ and $\mathbf{I}_m$) are cc-stateless. Furthermore, we assume that $\mathbf{F}_i(\cdot)$ is efficiently implementable for all $i = 1, \ldots, m-1$,[20] and the corresponding (short) random variable S_i is drawn from the set $\mathcal{S}_i$. Also, we let $\mathbf{E}(\cdot)$ be construction restricting access to $\mathbf{F}_i$ and $\mathbf{I}_i$. Finally, for $i = 1, \ldots, m$, and for $s_1 \in \mathcal{S}_1, \ldots, s_{i-1} \in \mathcal{S}_{i-1}$ we define

$$\mathbf{C}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot) := \mathbf{C}(\mathbf{F}_1(s_1), \ldots, \mathbf{F}_{i-1}(s_{i-1}), \cdot, \mathbf{F}_{i+1}, \ldots, \mathbf{F}_m)$$

and consider the following two properties:

(i) For all $i = 1, \ldots, m-1$ (the property is not necessary for $i = m$) and all $s_1 \in \mathcal{S}_1, \ldots, s_{i-1} \in \mathcal{S}_{i-1}$, the ideal system $\mathbf{I}_i$ is η-self-independent under the construction $\mathbf{C}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$ for some small function η.
(ii) For all $i = 1, \ldots, m$ and $s_1 \in \mathcal{S}_1, \ldots, s_{i-1} \in \mathcal{S}_{i-1}$, there exists a construction $\mathbf{T}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$ with the property that for independent instances $\mathbf{T}_1(\cdot), \ldots, \mathbf{T}_\lambda(\cdot)$ and $\mathbf{C}_1(\cdot), \ldots, \mathbf{C}_\lambda(\cdot)$ of $\mathbf{T}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$ and $\mathbf{C}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$, respectively, and all compatible systems $\mathbf{S}$,

$$\mathbf{T}_1(\mathbf{E}(\mathbf{S})) \| \cdots \| \mathbf{T}_\lambda(\mathbf{E}(\mathbf{S})) \equiv \mathbf{C}_1(\mathbf{S}) \| \cdots \| \mathbf{C}_\lambda(\mathbf{S}).$$

We define t_{T_i} as the maximal complexity (taken over all $s_1, \ldots, s_{i-1}$) for implementing the construction $\mathbf{T}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$.

In the following, we define $\lambda := \left(\frac{4m}{\gamma}\right)^2 \cdot \ln\left(\frac{4m}{\gamma}\right)$, for understood m and γ.

Theorem 3 (Strong Product Theorem). *Let $q > 0$ and $\mathbf{C}(\cdot)$ be as above satisfying conditions* (i) *and* (ii)*, and assume that upon q queries, $\mathbf{C}(\cdot)$ makes at most q_i queries to the i-th subsystem. Then, for all $t, \gamma > 0$,*

[20] While the same techniques as in the proof of Theorem 1 could be used to address general cc-stateless systems where $\mathbf{F}(\cdot)$ is not necessarily efficient, this will not be necessary for our applications.

$$\Delta_{t,q}(\mathbf{C}(\mathbf{F}_1,\ldots,\mathbf{F}_m),\mathbf{C}(\mathbf{I}_1,\ldots,\mathbf{I}_m)) \leq \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\mathbf{E}(\mathbf{F}_i),\mathbf{E}(\mathbf{I}_i)) + \sum_{i=1}^{m-1} \eta(q_i,\lambda)+\gamma,$$

where $t'_i := \lambda \cdot (t + \mathcal{O}(t_{T_i}(q_i)))$ *and* $q'_i := \lambda \cdot q_i$ *for all* $i = 1,\ldots,m-1$*, whereas* $t'_m := t + \mathcal{O}(t_{T_m}(q))$ *and* $q'_m := q_m$*.*

The proof of Theorem 3 is deferred to the full version of this paper. It abstracts and generalizes the proof technique used by Myers [27] (which was in turn based on Levin's proof of the XOR-lemma [20,10]).

4.2 Applications of the Strong Product Theorem

We present a number of new results which follow as simple applications of Theorem 3. Let $\mathbf{Q}_1,\ldots,\mathbf{Q}_m : \{0,1\}^n \to \{0,1\}^n$ be cc-stateless random permutations, and let $\mathbf{F}_1,\ldots,\mathbf{F}_m : \{0,1\}^n \to \{0,1\}^\ell$ be cc-stateless random functions. Furthermore, let $\mathbf{P} : \{0,1\}^n \to \{0,1\}^n$ and $\mathbf{R} : \{0,1\}^n \to \{0,1\}^\ell$ be a URF and URP, respectively. Assume that $\mathbf{Q}_i(s,x)$ (and $\mathbf{Q}_i^{-1}(s,y)$) and $\mathbf{F}_i(s,x)$ can be computed in time t_{Q_i} and t_{F_i}, respectively, for all s, x, and y.

Randomized Cascade of PRPs. The perhaps most surprising application is a *strong* product theorem for (two-sided) PRPs. We modify the (two-sided) cascade $\langle\mathbf{Q}_1\rangle \rhd \cdots \rhd \langle\mathbf{Q}_m\rangle$ by choosing two independent random offsets that are added to the inputs and the outputs, i.e., we consider $\langle\oplus_{Z_1}\rangle \rhd \langle\mathbf{Q}_1\rangle \rhd \cdots \rhd \langle\mathbf{Q}_m\rangle \rhd \langle\oplus_{Z_2}\rangle$ for two independent uniform n-bit strings Z_1, Z_2, where for some $z \in \{0,1\}^n$ the system $\langle\oplus_z\rangle$ is the bi-directional mapping which answers a forward query $(x,+)$ with $x \oplus z$ and a backward query $(y,-)$ with $y \oplus z$. The computational overhead is minimal compared to the regular cascade, and requires only additional storage for two n-bit strings (which are to be seen as part of the secret key).

Clearly the neutralizing property of the original cascade is preserved. Furthermore, using techniques from [22], we show in the full version that the construction satisfies condition (i) above with $\eta(q,\lambda) \leq q^2\lambda^2 2^{-n}$. Therefore, Theorem 3 (with $\mathbf{E}(\cdot)$ being the identity) yields the following result.

Corollary 3. *For all* $t,q,\gamma > 0$*, and independent uniform* n*-bit strings* Z_1, Z_2*,*

$$\Delta_{t,q}(\langle\oplus_{Z_1}\rangle \rhd \langle\mathbf{Q}_1\rangle \rhd \cdots \rhd \langle\mathbf{Q}_m\rangle \rhd \langle\oplus_{Z_2}\rangle, \langle\mathbf{P}\rangle) \leq \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\langle\mathbf{Q}_i\rangle,\langle\mathbf{P}\rangle) + \tfrac{mq^2\lambda^2}{2^n} + \gamma,$$

where $t'_i := \lambda \cdot \left(t + \mathcal{O}(q \cdot \sum_{j\neq i} t_{Q_j})\right)$ *and* $q'_i := \lambda \cdot q$ *for all* $i = 1,\ldots,m-1$*, whereas* $t'_m := t + \mathcal{O}(q \cdot \sum_{j=1}^{m-1} t_{Q_j})$ *and* $q'_m := q$*.*

The result can be used to obtain a δ^m-two-sided PRP from any δ-two-sided PRP. (Note that the η-dependent term is negligible for polynomial t,q and any γ which is the inverse of a polynomial.) It can be shown that the second random offset Z_2 is superfluous in the one-sided case.

Sum of Random-Input PRFs. The construction $\mathbf{K}(\mathbf{F}_1 \oplus \cdots \oplus \mathbf{F}_m)$ (i.e. the XOR of the functions accessed in a random-input attack) is clearly neutralizing (the ideal system being $\mathbf{K}(\mathbf{R})$). In the full version, we show that it also satisfies condition (i) with $\eta(q, \lambda) \leq \frac{q^2\lambda^2}{2} \cdot 2^{-n}$. Moreover, for all i and keys $s_1 \in \mathcal{S}_1, \ldots, s_{i-1} \in \mathcal{S}_{i-1}$, the appropriate construction $\mathbf{T}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$ generates random keys $S_{i+1}, \ldots, S_m$ and whenever invoked, it issues a query to $\mathbf{K}(\mathbf{S})$, obtaining (r, y), and outputs the pair

$$\Big(r, \bigoplus_{j=1}^{i-1} \mathbf{F}_i(s_i, r) \oplus y \oplus \bigoplus_{j=i+1}^{m} \mathbf{F}_j(S_j, r)\Big).$$

It is easy to see that these constructions satisfy property (ii), since $\mathbf{K}(\cdot)$ evaluates the given function at a fresh random input upon each invocation. Theorem 3 yields the following result.

Corollary 4. *For all $t, q, \gamma > 0$,*

$$\Delta_{t,q}(\mathbf{K}(\mathbf{F}_1 \oplus \cdots \oplus \mathbf{F}_m), \mathbf{K}(\mathbf{R})) \leq \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\mathbf{K}(\mathbf{F}_i), \mathbf{K}(\mathbf{R})) + \tfrac{(m-1)q^2\lambda^2}{2^{n+1}} + \gamma,$$

where $t'_i := \lambda\big(t + \mathcal{O}(q \cdot \sum_{j \neq i} t_{F_j})\big)$ and $q'_i := \lambda \cdot q$ for all $i = 1, \ldots, m-1$, whereas $t'_m := t + \mathcal{O}(q \cdot \sum_{j=1}^{m-1} t_{F_j})$ and $q'_m := q$.

The result holds for any other quasi-group operation. It is remarkable that XOR satisfies much stronger indistinguishability amplification properties under random-input attacks than under chosen-input attacks. This is particularly interesting, as a wide number of applications, such as secure symmetric message encryption, can efficiently be based on this weaker PRF notion (cf. [5,25]).

Randomized XOR of PRFs. The first product theorem for PRFs, due to Myers [27], considered the neutralizing composition $\mathbf{Z}_1(\mathbf{F}_1) \oplus \cdots \oplus \mathbf{Z}_m(\mathbf{F}_m)$ for independent instances of $\mathbf{Z}(\cdot)$. This result is directly implied by Theorem 3, which in fact also implies the same result for the construction $\mathbf{Z}(\mathbf{F}_1 \oplus \cdots \oplus \mathbf{F}_m)$ using the *same* offset for all invocations: As we show in the full version, both compositions satisfy property (i) with $\eta(q, \lambda) \leq \frac{q^2\lambda^2}{2} 2^{-n}$.

However, a major advantage of Myers' original construction (which was unobserved so far) is that independent instances of the construction can be simulated even when only given access to $\mathbf{Z}(\mathbf{S})$ (with $\mathbf{S} \in \{\mathbf{F}_i, \mathbf{R}\}$). The corresponding construction $\mathbf{T}^{(i)}_{s_1,\ldots,s_{i-1}}(\cdot)$ chooses independent instances $\mathbf{F}_{i+1}, \ldots, \mathbf{F}_m$, $\mathbf{Z}_1(\cdot), \ldots, \mathbf{Z}_{i-1}(\cdot), \mathbf{Z}_{i+1}(\cdot), \ldots, \mathbf{Z}_m(\cdot)$, and a random n-bit string Z, and on input x queries $x \oplus Z$ to $\mathbf{Z}(S)$, obtaining $y \in \{0,1\}^\ell$, and outputs

$$y \oplus \bigoplus_{j=1}^{i-1} \mathbf{Z}_j(\mathbf{F}_j(s_j))(x) \oplus \bigoplus_{j=i+1}^{m} \mathbf{Z}_j(\mathbf{F}_j)(x),$$

where $\mathbf{Z}_j(\mathbf{F}_j)(x)$ is the result of invoking the system $\mathbf{Z}_j(\mathbf{F}_j)$ on input x.

Once again, condition (ii) is easily verified by the fact that access through $\mathbf{Z}(\cdot)$ can be re-randomized by simply adding a fresh random offset to all inputs. Thus, Theorem 3 yields the following strengthened version of the main result of [27].

Corollary 5. *For all $t, q, \gamma > 0$, and for independent instances $\mathbf{Z}_1(\cdot), \ldots, \mathbf{Z}_m(\cdot)$ of $\mathbf{Z}(\cdot)$,*

$$\Delta_{t,q}(\mathbf{Z}_1(\mathbf{F}_1) \oplus \cdots \oplus \mathbf{Z}_m(\mathbf{F}_m), \mathbf{R}) \leq \prod_{i=1}^{m} \Delta_{t'_i,q'_i}(\mathbf{Z}(\mathbf{F}_i), \mathbf{Z}(\mathbf{R})) + \tfrac{(m-1)q^2\lambda^2}{2^{n+1}} + \gamma,$$

where $t'_i := \lambda\big(t + \mathcal{O}(q \cdot \sum_{j \neq i} t_{F_j})\big)$ and $q'_i := \lambda \cdot q$ for all $i = 1, \ldots, m-1$), whereas $t'_m := t + \mathcal{O}(q \cdot \sum_{j=1}^{m-1} t_{F_j})$ and $q'_m := q$.

The best advantage under $\mathbf{Z}(\cdot)$ can be significantly smaller than under direct access: Consider e.g. a good PRF with the additional property of outputting the zero string when evaluated at some fixed known input, regardless of the key.

References

1. Bellare, M., Impagliazzo, R., Naor, M.: Does parallel repetition lower the error in computationally sound protocols? In: FOCS 1997, pp. 374–383 (1997)
2. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
3. Canetti, R., Halevi, S., Steiner, M.: Hardness amplification of weakly verifiable puzzles. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 17–33. Springer, Heidelberg (2005)
4. Canetti, R., Rivest, R.L., Sudan, M., Trevisan, L., Vadhan, S.P., Wee, H.: Amplifying collision resistance: A complexity-theoretic treatment. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 264–283. Springer, Heidelberg (2007)
5. Damgård, I.B., Nielsen, J.B.: Expanding pseudorandom functions; or: From known-plaintext security to chosen-plaintext security. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 449–464. Springer, Heidelberg (2002)
6. Dodis, Y., Impagliazzo, R., Jaiswal, R., Kabanets, V.: Security amplification for interactive cryptographic primitives. In: TCC 2009. LNCS, vol. 5444, pp. 128–145 (2009)
7. Dwork, C., Naor, M., Reingold, O.: Immunizing encryption schemes from decryption errors. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 342–360. Springer, Heidelberg (2004)
8. Even, S., Goldreich, O.: On the power of cascade ciphers. ACM Trans. Comput. Syst. 3(2), 108–116 (1985)
9. Goldreich, O., Impagliazzo, R., Levin, L.A., Venkatesan, R., Zuckerman, D.: Security preserving amplification of hardness. In: FOCS 1990, pp. 318–326 (1990)
10. Goldreich, O., Nisan, N., Wigderson, A.: On Yao's XOR-lemma. Electronic Colloquium on Computational Complexity (ECCC) 2(50) (1995)
11. Haitner, I., Harnik, D., Reingold, O.: On the power of the randomized iterate. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 22–40. Springer, Heidelberg (2006)
12. Halevi, S., Rabin, T.: Degradation and amplification of computational hardness. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 626–643. Springer, Heidelberg (2008)
13. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
14. Herzberg, A.: On tolerant cryptographic constructions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 172–190. Springer, Heidelberg (2005)

15. Holenstein, T.: Key agreement from weak bit agreement. In: STOC 2005, pp. 664–673 (2005)
16. Holenstein, T., Renner, R.: One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 478–493. Springer, Heidelberg (2005)
17. Hopper, N., Molnar, D., Wagner, D.: From weak to strong watermarking. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 362–382. Springer, Heidelberg (2007)
18. Impagliazzo, R.: Hard-core distributions for somewhat hard problems. In: FOCS 1995, pp. 538–545 (1995)
19. Impagliazzo, R., Jaiswal, R., Kabanets, V.: Chernoff-type direct product theorems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 500–516. Springer, Heidelberg (2007)
20. Levin, L.A.: One way functions and pseudorandom generators. Combinatorica 7(4), 357–363 (1987)
21. Luby, M., Rackoff, C.: Pseudo-random permutation generators and cryptographic composition. In: STOC 1986, pp. 356–363 (1986)
22. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
23. Maurer, U., Massey, J.L.: Cascade ciphers: The importance of being first. Journal of Cryptology 6(1), 55–61 (1993)
24. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
25. Maurer, U., Sjödin, J.: A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 498–516. Springer, Heidelberg (2007)
26. Myers, S.: On the development of block-ciphers and pseudo-random function generators using the composition and XOR operators. Master's thesis, University of Toronto (1999)
27. Myers, S.: Efficient amplification of the security of weak pseudo-random function generators. Journal of Cryptology 16, 1–24 (2003)
28. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. Journal of Computer and System Sciences 58(2), 336–375 (1999)
29. Pass, R., Venkitasubramaniam, M.: An efficient parallel repetition theorem for Arthur-Merlin games. In: STOC 2007, pp. 420–429 (2007)
30. Pietrzak, K., Wikström, D.: Parallel repetition of computationally sound protocols revisited. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 86–102. Springer, Heidelberg (2007)
31. Shaltiel, R., Viola, E.: Hardness amplification proofs require majority. In: STOC 2008, pp. 589–598 (2008)
32. Vaudenay, S.: Provable security for block ciphers by decorrelation. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 249–275. Springer, Heidelberg (1998)
33. Vaudenay, S.: Adaptive-attack norm for decorrelation and super-pseudorandomness. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 49–61. Springer, Heidelberg (2000)
34. Wullschleger, J.: Oblivious-transfer amplification. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 555–572. Springer, Heidelberg (2007)
35. Yao, A.C.: Theory and applications of trapdoor functions. In: FOCS 1982, pp. 80–91 (1982)

Merkle Puzzles Are Optimal — An $O(n^2)$-Query Attack on Any Key Exchange from a Random Oracle

Boaz Barak[⋆] and Mohammad Mahmoody-Ghidary[⋆⋆]

Department of Computer Science, Princeton University
{boaz,mohammad}@cs.princeton.edu

Abstract. We prove that every key exchange protocol in the random oracle model in which the honest users make at most n queries to the oracle can be broken by an adversary making $O(n^2)$ queries to the oracle. This improves on the previous $\tilde{\Omega}(n^6)$ query attack given by Impagliazzo and Rudich (STOC '89), and answers an open question posed by them. Our bound is optimal up to a constant factor since Merkle (CACM '78) gave a key exchange protocol that can easily be implemented in this model with n queries and cannot be broken by an adversary making $o(n^2)$ queries.

1 Introduction

In the 1970's Diffie, Hellman, and Merkle began to challenge the accepted wisdom that two parties cannot communicate confidentially over an open channel without first exchanging a secret key using some secure means. The first such protocol (at least in the open scientific community) was designed by Merkle in 1974 (although only published in 1978 [1]). Merkle's protocol allows two parties Alice and Bob to agree on a random number k that will not be known to an eavesdropping adversary Eve. It is described in Fig. 1.

One problem with Merkle's protocol is that its security was only analyzed in the random oracle model which does not necessarily capture security when instantiated with a cryptographic one-way or hash function [4]. Recently, Biham, Goren and Ishai [3] took a step towards resolving this issue by providing a security analysis for Merkle's protocol under the concrete complexity assumption of existence of exponentially hard one-way functions. In particular, they proved that assuming there exist a one-way function that cannot be inverted with probability more than $2^{-\alpha n}$ by adversaries running in time $2^{\alpha n}$ for $\alpha \geq 1/2 - \delta$, there is a key exchange protocol in which Alice and Bob run in time n but any adversary whose running time is at most $n^{2-10\delta}$ has $o(1)$ chance of finding the

⋆ Supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797, US-Israel BSF grant 2004288 and Packard and Sloan fellowships.

⋆⋆ Supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797.

Merkle's Key Exchange Protocol

Let n be the security parameter. All parties have access to oracle to a function $H : \{0,1\}^\ell \to \{0,1\}^\ell$ chosen at random, where $\ell \gg \log n$. The protocol operates as follows:

1. Alice chooses $10n$ random numbers $x_1, \ldots, x_{10n}$ in $[n^2]$ and sends $a_1, \ldots, a_{10n}$ to Bob where $a_i = H(x_i)$ (embed $[n^2]$ in $\{0,1\}^\ell$ in some canonical way).
2. Bob chooses $10n$ random numbers $y_1, \ldots, y_{10n}$ in $[n^2]$ and sends $b_1, \ldots, b_{10n}$ to Alice where $b_j = H(x_j)$.
3. With at least 0.9 probability, there will be at least one "collision" between Alice's and Bob's messages: a pair i, j such that $a_i = b_j$. Alice and Bob choose the lexicographically first such pair, and Alice sets $s_A = x_i$ as her secret, and Bob sets $s_B = y_j$ as his secret. If no collision occurred they will not choose any secret. Note that assuming $2^\ell \gg n^4$, H will be one to one on $[n^2]$ with very high probability and hence $H(x_i) = H(y_j)$ implies $x_i = y_j$.

To analyze the protocol one shows that the collision is distributed uniformly in $[n^2]$ and deduces that an adversary Eve that makes $o(n^2)$ queries to the oracle will find the secret with $o(1)$ probability.

Fig. 1. Merkle's key exchange protocol [1][1]

secret. But the most serious issue with Merkle's protocol is that it only provides a *quadratic* gap between the running time of the honest parties and the adversary. Fortunately, not too long after Merkle's work, Diffie and Hellman [5] and later Rivest, Shamir, and Adleman [6] gave constructions for key exchange protocols that are conjectured to have *super-polynomial* (even subexponential) security. But because these and later protocols are based on certain algebraic computational problems, and so could perhaps be vulnerable to unforseen attacks using this algebraic structure, it remained an important question to show whether there exist key exchange protocols with superpolynomial security that use only a random oracle.[2] The seminal paper of Impagliazzo and Rudich [8] answered this question negatively by showing that every key exchange protocol using n queries in the random oracle model can be broken by an adversary asking

[1] Merkle described his protocol using "puzzles" that can be implemented via some ideal cryptographic primitive; we describe the protocol in the case that the puzzles are implemented by a random oracle. We remark that in Merkle's original protocol Bob will try different random queries $y_1, y_2, \ldots$ without sending them to Alice until he finds y_j such that $f(y_j) \in \{a_1, \ldots, a_{10n}\}$ and send j — the index of the "puzzle" a_j — to Alice. The Protocol of Fig. 1 is a *symmetric* version of Merkle's protocol, and is similar to the protocol of [2] in the bounded storage model; see also discussion in [3].

[2] This is not to be confused with some more recent works such as [7], that combine the random oracle model with assumptions on the intractability of other problems such as factoring or the RSA problem to obtain more efficient cryptographic constructions.

$O(n^6 \log n)$ queries.[3] Since a random oracle is in particular a one-way function (with high probability), this implied that there is no construction of a key exchange protocol based on a one-way function with a proof of super-polynomial security that is of the standard black-box type (i.e., a proof that transforms an adversary breaking the protocol into an inversion algorithm for the one-way function that only uses the adversary and the function as black boxes). Indeed, that was the motivation behind their result.

Question and Motivation. Impagliazzo and Rudich [8, Sect. 8] mention as an open question (which they attribute to Merkle) to find out whether their attack can be improved to $O(n^2)$ queries (hence showing the optimality of Merkle's protocol in the random oracle model) or there exist key exchange protocols in the random oracle model with $\omega(n^2)$ security. Beyond just being a natural question, it also has some practical and theoretical motivations. The practical motivation is that protocols with sufficiently large polynomial gap could be secure enough in practice — e.g., a key exchange protocol taking 10^9 operations to run and $(10^9)^6 = 10^{54}$ operations to break could be good enough for many applications.[4] In fact, as was argued by [3], as technology improves and honest users can afford to run more operations, such polynomial gaps only become more useful. Thus if known algebraic key exchange protocols were broken, one might look to polynomial-security protocol such as Merkle's for an alternative. Another motivation is theoretical— Merkle's protocol has very limited interaction (consisting of one round in which both parties simultaneously broadcast a message) and in particular it implies a public key encryption scheme. It is natural to ask whether more interaction can help achieve some polynomial advantage over this simple protocol. A third, less direct motivation comes from quantum computing. In one scenario in which some algebraic key exchange protocols will be broken— the construction of practical quantum computers— Merkle's protocol will also fail to offer non-trivial security due to Grover's search algorithm [9]. Our results below suggest (though do not prove) that Merkle's protocol may be optimal in this setting also, and so there may not exist a fully classical key-exchange protocol based on a one-way function with a black-box proof of super-linear security for quantum adversaries. We note that using quantum communication there is an *information theoretically* secure key-exchange protocol [10], and moreover, very recently Brassard and Salvail [11] (independently observed by [3]) gave a quantum version of Merkle's protocol, showing that if Alice and Bob can use quantum computation (but classical communication), to obtain a key-exchange

[3] More accurately, [8] gave an $O(m^6 \log m)$-query attack where m is the maximum of the number of queries n and the number of communication rounds, though we believe their analysis could be improved to an $O(n^6 \log n)$-query attack. For the sake of simplicity, when discussing [8]'s results we will assume that $m = n$, though for our result we do not need this assumption.

[4] Of course, these numbers are just an example and in practical applications the constant terms will make an important difference. We note though that the above constants are not ruled out by [8]'s attack, but are ruled out by our attack (taking number of operations to mean the number of calls to the oracle).

protocol with super-linear (i.e., $n^{3/2}$) security in the random oracle model against quantum adversaries.

Our Result. In this work we answer the above question of [8], by showing that every protocol in the random oracle model where Alice and Bob make n oracle queries can be broken with high probability by an adversary making $O(n^2)$ queries. That is, we prove the following:

Theorem 1. *Let Π be a two-party protocol in the random oracle model such that when executing Π the two parties Alice and Bob make at most n queries each, and their outputs are identical with probability at least ρ. Then for every $0 < \delta < 1$, there is an adversary Eve making $(\frac{16n}{\delta})^2$ queries to the oracle whose output agrees with Bob's output with probability at least $\rho - \delta$.*

To the best of our knowledge, no better bound than the $\tilde{O}(n^6)$-query attack of [8] was previously known even in the case where one does not assume the one-way function is a random oracle (hence making the task of proving a negative result easier). We note that similarly to previous black-box separation results, our adversary can be implemented efficiently in a relativized world where $\mathbf{P} = \mathbf{NP}$.

Correction of Error: A previous version of this manuscript [12] posted on the Arxiv claimed a different proof of the same result. However, we have found a bug in that proof— see the full version of this paper for more details. In fact the current proof is quite different from the one claimed in [12]. In [12] we also claimed an extension of Theorem 1 to the case of protocols with an oracle to a *random permutation* (i.e., a random one-to-one function R from $\{0,1\}^*$ to $\{0,1\}^*$ such that $|R(x)| = |x|$ for every $x \in \{0,1\}^*$). We do not know of an extension of the current proof to this model, beyond the observation of [8] that any m-query attack in the random oracle model translates into an $O(m^2)$-query attack in the random permutation model. Hence our results imply an $O(n^4)$-query attack in the latter model, improving on the previous $\tilde{O}(n^{12})$ attack of [8].

We also note that shortly after we posted the manuscript [12], Sotakova [13] posted an independently obtained weaker result, showing that protocols with only one round of interaction (each party sends one message) and non-adaptive queries can achieve at most $O(n^2)$ security. In contrast, as in the work of [8], in this paper we allow protocols where the parties' choice of queries is adaptive and they can use an arbitrary polynomial number of interaction rounds.[5] The one-round case seems to be simpler, and in particular the bug found in our previous proof does not apply to that case.

2 Our Techniques

The main technical challenge in proving such a result is the issue of *dependence* between the executions of the two parties Alice and Bob in a key exchange

[5] In fact, because we count only the number of *oracle queries* made by the honest parties, we can even allow a super-polynomial number of rounds.

protocol. The presence of the random oracle allows Alice and Bob to correlate their executions even without communicating (which is indeed the reason that Merkle's protocol achieves non-trivial security). Dealing with such correlations is the cause of the technical complexity in both our work and the previous work of Impagliazzo and Rudich [8]. We handle this issue in a different way than [8]. On a very vague high level our approach can be viewed as using more information about the structure of these correlations than [8] did. This allows us to analyze a more efficient attacking algorithm, that is more frugal with the number of queries it uses than the attacker of [8]. Below we provide a more detailed (though still high level) exposition of our technique and its relation to [8]'s technique.

2.1 Comparison with [8]

We now review [8]'s attack and outline of analysis, and particularly the subtle issue of *dependence* between Alice and Bob that arises in both their work and ours. The main novelty of our work is the way we deal with this issue, which is different from the approach of [8]. We believe that this review of [8]'s analysis and the way it compares to ours can serve as a useful introduction to our actual proof. However, no result of this section is used in the later sections, and so the reader should feel free at any time to skip ahead to Sect. 3 and 4 that contain our actual attack and its analysis.

Consider a protocol that consists of n rounds of interaction, where each party makes exactly one oracle query before sending its message. [8] called protocols of this type "normal-form protocols" and gave an $\tilde{O}(n^3)$ attack against them (their final result was obtained by transforming every protocol into a normal-form protocol with a quadratic loss of efficiency). Even though without loss of generality the attacker Eve of a key exchange protocol can defer all of her computation till after the interaction between Alice and Bob is finished, it is conceptually simpler in both [8]'s case and ours to think of the attacker Eve as running concurrently with Alice and Bob. In particular, the attacker Eve of [8] performed the following operations after each round i of the protocol:

- If the round i is one in which Bob sent a message, then at this point Eve samples $1000n \log n$ random executions of Bob from the distribution $\mathcal{D}$ of Bob's executions that are consistent with the information that Eve has at that moment (communication transcript and previous oracle answers). That is, Eve samples a uniformly random tape for Bob and uniformly random query answers subject to being consistent with Eve's information. After each time that she samples an execution, Eve asks the oracle all the queries asked during this execution and records the answers. (Generally, the true answers will not be the same answers as the one Eve guessed when sampling the execution.)
- Similarly, if the round i is one in which Alice sent a message then Eve samples $1000n \log n$ executions of Alice and makes the corresponding queries.

Overall Eve will sample $\tilde{O}(n^2)$ executions making a total of $\tilde{O}(n^3)$ queries. It's not hard to see that as long as Eve learns all of the *intersection queries* (queries asked by both Alice and Bob during the execution) then she can recover the shared secret with high probability. Thus the bulk of [8]'s analysis was devoted to showing the following statement, denoted below by **(*)**: *With probability at least* 0.9 *Eve never fails, where we say that Eve* fails *at round i if the query made in this round by, say, Alice was asked previously by Bob but not by Eve.*

2.2 The Issue of Independence

At first look, it may seem that one could easily prove **(*)**. Indeed, **(*)** will follow by showing that at any round i, the probability that Eve fails in round i *for the first time* is at most $1/(10n)$. Now all the communication between Alice and Bob is observed by Eve, and if no failure has yet happened then Eve has also observed all the intersection queries so far. Because the answers for non-intersection queries are completely random and independent from one another it seems that Alice has no more information about Bob than Eve does, and hence if the probability that Alice's query q was asked before by Bob is more than $1/(10n)$ then this query q has probability at least $1/(10n)$ to appear in each one of Eve's sampled executions of Bob. Since Eve makes $1000n \log n$ such samples, the probability that Eve misses q would be bounded by $(1-\frac{1}{10n})^{1000n \log n} \ll 1/(10n)$.

When trying to make this intuition into a proof, the assumption that Eve has as much information about Bob as Alice does translates to the following statement: conditioned on Eve's information, the distributions of Alice's view and Bob's view are *independent* from one another.[6] Indeed, if this statement was true then the above paragraph could be easily translated into a proof that [8]'s attacker is successful, and it wouldn't have been hard to optimize this attacker to achieve $O(n^2)$ queries. Alas, this statement is false. Intuitively the reason is the following: even the fact that Eve has not missed any intersection queries is some non-trivial information that Alice and Bob share and creates dependence between them.[7]

Impagliazzo and Rudich [8] dealt with this issue by a "charging argument" (see also Remark 2 below), where they showed that such dependence can be charged in a certain way to one of the executions sampled by Eve, in a way that at most n samples can be charged at each round (and the rest of Eve's

[6] Readers familiar with the setting of communication complexity may note that this is analogous to the well known fact that conditioning on any transcript of a 2-party communication protocol results in a product distribution (i.e., combinatorial rectangle) over the inputs. However, things are different in the presence of a random oracle.

[7] As a simple example for such dependence consider a protocol where in the first round Alice chooses x to be either the string 0^n or 1^n at random, queries the oracle H at x and sends $y = H(x)$ to Bob. Bob then makes the query 1^n and gets $y' = H(1^n)$. Now even if Alice chose $x = 0^n$ and hence Alice and Bob have no intersection queries, Bob can find out the value of x just by observing that $y' \neq y$. Still, an attacker must ask a non-intersection query such as 1^n to know if $x = 0^n$ or $x = 1^n$.

samples are distributed correctly as if the independence assumption was true). This argument inherently required sampling at least n executions (each of n queries) per round, hence resulting in an $\Omega(n^3)$ attack.

2.3 Our Approach

We now describe our approach and how it differs from the previous proof of [8]. The discussion below is somewhat high level and vague, and glosses over some important details. Again, the reader is welcome to skip ahead at any time to Sect. 3 that contains the full description of our attack, and does not depend on this section in any way.

Our attacking algorithm follows the same general outline, but has two important differences from the attacker of [8]:

1. One *quantitative* difference is that while our attacker Eve also computes a distribution $\mathcal{D}$ of possible executions of Alice and Bob conditioned on her knowledge, she does *not* sample from $\mathcal{D}$ full executions and then ask the arising queries. Rather, she computes whether there is any *heavy query*— a string $q \in \{0,1\}^*$ that has probability more than, say, $1/(100n)$ of being queried in $\mathcal{D}$— and makes only such heavy queries.

   Intuitively, since Alice and Bob make at most $2n$ queries, the total expected number of heavy queries (and hence the query complexity of Eve) is bounded by $O(n^2)$. The actual analysis is more involved since the distribution $\mathcal{D}$ keeps changing as Eve learns more information through the messages she observes and query answers she receives. We omit the details in this high-level overview.
2. The *qualitative* difference between the two attackers is that we do not consider the same distribution $\mathcal{D}$ that was considered by [8]. Their attacker to some extent "pretended" that the conditional distributions of Alice and Bob are independent from one another. In contrast, we define our distribution $\mathcal{D}$ to be the *real* distribution of Alice and Bob, where there could be dependencies between them. Thus to sample from our distribution $\mathcal{D}$ one would need to sample a *pair* of executions of Alice and Bob (random tapes and oracle answers) that are *jointly consistent* with one another and Eve's current knowledge. Another (less important) point is that the distribution $\mathcal{D}$ computed by Eve at each point in time will be conditioned not only on Eve's knowledge so far, but also on the event that she has not failed until this point.

The main challenge in the analysis is to prove that the attack is *successful*, that is that the statement **(*)** above holds, and in particular that the probability of failure at each round (or more generally, at each query of Alice or Bob) is bounded by, say, $1/(10n)$. Once again, things would have been easy if we knew that the distribution $\mathcal{D}$ of the possible executions of Alice and Bob conditioned on Eve's knowledge (and not having failed so far) is a *product distribution*, and hence Alice has no more information on Bob than Eve has. While this is not generally true, we show that in our attack this distribution is *close to being a product distribution*, in a precise sense we define below.

At any point in the execution, fix Eve's current information about the system and define a bipartite graph G whose left-side vertices correspond to possible executions of Alice that are consistent with Eve's information and right-side vertices correspond to possible executions of Bob consistent with Eve's information. We put an edge between two executions A and B if they are consistent with one another and moreover if they do not represent an execution in which Eve *failed* prior to this point (i.e., there is no intersection query that is asked in both executions A and B but not by Eve). The distribution $\mathcal{D}$ that our attacker Eve considers can be thought of as choosing a random edge in the graph G. (Note that the graph G and the distribution $\mathcal{D}$ change at each point that Eve learns some new information about the system.) If G was the complete bipartite clique then $\mathcal{D}$ would be a product distribution. What we show is that G is *dense* in the sense that each vertex is connected to most of the vertices on the other side. We show that this implies that Alice's probability of hitting a query that Bob asked before is at most twice the probability that Eve does so if she chooses the most likely query based on her knowledge.

The bound on the degree is obtained by showing that G can be represented as a *disjointness graph*, where each vertex u is associated with a set $S(u)$ (from an arbitrarily large universe) and there is an edge between a left-side vertex u and a right-side vertex v if and only if $S(u) \cap S(v) = \emptyset$.[8] The definition of the graph G implies that $|S(u)| \leq n$ for all vertices u. The definition of our attacking algorithm implies that the distribution obtained by picking a random edge $\{u, v\}$ and outputting $S(u) \cup S(v)$ is *light*, in the sense that there is no element q in the universe that has probability more than $1/(10n)$ of being contained in a set chosen from this distribution. We show that these properties together imply that each vertex is connected to most of the vertices on the other side, and so G is close to being a complete bipartite graph.

Remark 2 (Comparison with [8]). One can also phrase the analysis of [8] in terms of a similar bipartite graph. Their argument involved fixing, say, Alice's execution which corresponds to fixing a left-side vertex u. As we noted above, if the degree of u is high (e.g., u is connected to most of the right side) then independence approximately holds and hence the probability that [8]'s attacker fails at this point is less than $1/(10n)$. The crucial component of [8]'s analysis was their observation that if the degree of u is low, then by taking a random vertex v on the right side and making all queries in the corresponding execution to v, one is likely to make progress in the sense that we learn a new query made in the execution corresponding to u. Now there are at most n new queries to learn, and hence if we sample much more than n queries then in most of them we're in the high degree case. This potential/charging argument inherently requires sampling *all* queries of the execution, rather than only the heavy ones, hence incurring a cost of at least n^2 queries *per round* or n^3 queries total.

[8] The set $S(u)$ will correspond to the queries that are made in the execution corresponding to u but *not* made by Eve.

3 Our Attacker

We consider a key exchange protocol Π in which Alice and Bob first toss coins r_A and r_B and then run Π using access to a random oracle H that is a random function from $\{0,1\}^\ell$ to $\{0,1\}^\ell$ for some $\ell \in \mathbb{N}$. We assume that the protocol proceeds in some finite number of rounds, and no party asks the same query twice. In round k, if k is odd then Alice makes some number of queries and sends a message to Bob (and then Eve asks some oracle queries), and if k is even then Bob makes some queries and sends a message to Alice (and then Eve asks some oracle queries). At the end of the protocol Alice obtains an output string s_A and Bob obtains an output string s_B. We assume that there is some constant $\rho > 0$ such that $\Pr[s_A = s_B] \geq \rho$, where the probability is over the coin tosses of Alice and Bob and the randomness of the oracle. We will establish Theorem 1 by proving that an attacker can make $O(n^2)$ queries to learn s_B with probability arbitrarily close to ρ.

In this section we describe an attacking algorithm that allows Eve to find a set of size $O(n^2)$ that contains all the queries asked by Alice and Bob in the random oracle model. This attack is analyzed in Sect. 4 to show that it is successful in finding all intersection queries and is efficient (i.e., will not ask more than $O(n^2)$ many queries). As was shown by Impagliazzo and Rudich, it not hard to use this set to obtain the actual secret.

3.1 Attacking Algorithm

We start by showing that an attacker can find all the *intersection queries* (those asked by both Alice and Bob) with high probability. It turns out that this is the main step in showing that an attacker can find the secret with high probability.

Theorem 3. *Let Π be a key exchange protocol in the random oracle model in which Alice and Bob ask at most n oracle queries each. Then for every $0 < \delta < 1$ there is an adversary Eve who has access to the messages sent between Alice and Bob and asks at most $(\frac{13n}{\delta})^2$ number of queries such that Eve's queries contain all the intersection queries of Alice and Bob with probability at least $1 - \delta$.*

Letting $\epsilon = \delta/13$, our attack can be described in one sentence as follows:

As long as there exists a string q such that conditioned on Eve's current knowledge and assuming that no intersection query was missed so far, the probability that q was asked in the past (by either Alice or Bob) is at least ϵ/n, Eve makes the query q to the oracle.

To describe the attack more formally, we need to introduce some notation. We fix n to be the number of oracle queries asked by Alice and Bob and assume without loss of generality that all the queries are of length $\ell = \ell(n)$ for some $\ell \in \mathbb{N}$. We will make the simplifying assumption that the protocol is in *normal form*— that is, at every round of the protocol Alice or Bob make exactly one query to the oracle (and hence there are $2n$ rounds). Later in Section 5 we will show how our analysis extends to protocols that are not of this form. Below and throughout the paper, we often identify a distribution $\mathcal{D}$ with a random variable distributed according to $\mathcal{D}$.

Executions and the Distribution $\mathcal{EXEC}$. A (full) *execution* of Alice, Bob, and Eve can be described by a tuple (r_A, r_B, H) where r_A denotes Alice's random tape, r_B denotes Bob's random tape, and H is the random oracle (note that Eve is deterministic). We denote by $\mathcal{EXEC}$ the distribution over (full) executions that is obtained by running the algorithms for Alice, Bob and Eve with uniformly chosen random tapes and a random oracle. A *partial execution* is an execution truncated at a certain point in time (that is, the transcripts contain only the oracle answers for queries that are asked up to that point). For any partial execution we denote by M the sequence of messages sent between Alice and Bob till that moment, and denote by I the set of oracle query/answer pairs known to Eve. We define *Alice's view* in the execution to be the tuple $A = (r_A, H_A, M)$ where r_A are Alice's coins and H_A is the concatenation of oracle answers to Alice's queries. Similarly Bob's view is the tuple $B = (r_B, H_B, M)$. Below we will only consider Alice's and Bob's view conditioned on a fixed value of M and hence we drop M from these tuples and let $A = (r_A, H_A)$ and $B = (r_B, H_B)$.

The Distribution $\mathcal{EXEC}(M, I)$. For $M = [m_1, \ldots, m_i]$ a sequence of i messages, and I a set of query/answer pairs, we denote by $\mathcal{EXEC}(M, I)$ the distribution over the views (A, B) of Alice and Bob in partial executions up to the point in the system in which the i^{th} message is sent (by Alice or Bob), where the transcript of messages equals M and the set of query/answer pairs that Eve learns equals I. For every (M, I) that have nonzero probability to occur in the protocol, the distribution $\mathcal{EXEC}(M, I)$ can be sampled by first sampling (r_A, r_B, H) at random conditioned on being consistent with (M, I) and then deriving from this tuple Alice's and Bob's views: $A = (r_A, H_A)$ and $B = (r_B, H_B)$.[9]

The Event $\mathsf{Good}(M, I)$ and the Distribution $\mathcal{GEXEC}(M, I)$. The event $\mathsf{Good}(M, I)$ is defined as the event over $\mathcal{EXEC}(M, I)$ that all the intersection queries asked by Alice and Bob during the partial execution are in I. More formally let $Q(A)$ (resp. $Q(B)$) be the set of queries asked by Alice (resp. Bob) which are specified by Alice's view A (resp. Bob's view B). Therefore $\mathsf{Good}(M, I)$ is the same as $Q(A) \cap Q(B) \subset Q(I)$ where $Q(I)$ is the set of queries of I (note that I is a set of query/answswer *pairs*). We define the distribution $\mathcal{GEXEC}(M, I)$ to be the distribution $\mathcal{EXEC}(M, I)$ conditioned on $\mathsf{Good}(M, I)$.

Eve's Algorithm. The attacker Eve's algorithm is specified as follows. It is parameterized by some constant $0 < \epsilon < 1/10$. At any point in the execution, if

[9] Note that we can verify that the pair (M, I) has nonzero probability to occur in the protocol by simulating Eve's algorithm on the transcript M, checking that whenever Eve makes a query, this query is in I, in which case we feed Eve with the corresponding answer (and verifying at the end that there are no "extra" queries in I not asked by Eve). However in our attack the pair (M, I) will always be generated by running the actual protocol and so we won't need to run such checks.

M is the sequence of messages Eve observed so far and I is the query/answer pairs she learned so far, Eve computes for every $q \in \{0,1\}^\ell$ the probability p_q that q appears as a query in a random execution in $\mathcal{GEXEC}(M, I)$. If $p_q > \epsilon/n$ then Eve asks q from the oracle and adds q and its answer to I. (If there is more than one such q then Eve asks the lexicographically first one.) Eve continues in this way until there is no additional query she can ask, at which point she waits until she gets new information (i.e., observes a new message sent between Alice and Bob).

Note that Eve's algorithm above may ask much more than n^2 queries. However, we will show that the probability that Eve asks more than n^2/ϵ^2 queries is bounded by $O(\epsilon)$, and hence we can stop Eve after asking this many queries without changing significantly her success probability.

4 Analysis of Attack: Proof of Theorem 3

We now go over the proof of Theorem 3. For $i \in [2n]$, define the event Fail_i to be the event that the query made at the i^{th} round is an intersection query but is not contained in the set I of query/answer pairs known by Eve, and moreover that this is the first query satisfying this condition. Let the event $\mathsf{Fail} = \bigvee_i \mathsf{Fail}_i$ be the event that at some point an intersection query is missed by Eve, and let the event Long be that Eve makes more than n^2/ϵ^2 queries. By setting $\epsilon = \delta/13$ and stopping Eve after n^2/ϵ^2 queries, Theorem 3 immediately follows from the following two lemmas:

Lemma 4 (Attack is successful). *For every i, $\Pr_{\mathcal{EXEC}}[\mathsf{Fail}_i] \le \frac{3\epsilon}{2n}$. Therefore by the union bound, $\Pr[\mathsf{Fail}] \le 3\epsilon$.*

Lemma 5 (Attack is efficient). $\Pr_{\mathcal{EXEC}}[\mathsf{Long}] \le 10\epsilon$.

4.1 Success of Attack: Proof of Lemma 4

Lemma 4 follows from the following stronger result which is the main technical lemma of our paper:

Lemma 6. *Let i be even and let $B = (r_B, H_B)$ be some fixing of Bob's view in an execution up to the i^{th} message sent by him, and let M, I be some fixing of the messages exchanged and query/answer pairs learned by Eve in this execution such that $\Pr_{\mathcal{EXEC}(M,I)}[\mathsf{Good}(M,I) \mid B] > 0$. Then it holds that $\Pr_{\mathcal{GEXEC}(M,I)}[\mathsf{Fail}_i \mid B] \le \frac{3\epsilon}{2n}$. That is, the probability that Fail_i happens is at most $\frac{3\epsilon}{2n}$ conditioning on Eve's information equalling (M, I), Bob's view of the execution equalling B and $\mathsf{Good}(M, I)$.*

Proof (of Lemma 4 from Lemma 6.). Lemma 6 implies that in particular for every even i, $\Pr_{\mathcal{EXEC}}[\mathsf{Fail}_i \mid \mathsf{Good}_i] \le \frac{3\epsilon}{2n}$, where Good_i denotes the event $\mathsf{Good}(M, I)$ where M, I are Eve's information just before the i^{th} round. But since Fail_i is the event that Eve fails at round i *for the first time*, Fail_i implies Good_i and hence

$\Pr_{\mathcal{EXEC}}[\mathsf{Fail}_i] \le \Pr_{\mathcal{EXEC}}[\mathsf{Fail}_i \mid \mathsf{Good}_i]$, establishing the statement of Lemma 4 for every even i. By symmetry, the analog of Lemma 6 for odd i also holds with the roles of Alice and Bob reversed, completing the proof for all i.

Proof of Lemma 6

Product Characterization. Lemma 6 would be easy if the distribution $\mathcal{GEXEC}(M, I)$ would have been a *product distribution*, with the views of Alice and Bob independent from one another. Roughly speaking this is because in this case Bob has no more information than Eve on the queries Alice made in the past, and hence also from Bob's point of view, no query is more probable than ϵ/n to have been asked by Alice. Unfortunately this is not the case. However, we can show that the distribution $\mathcal{GEXEC}(M, I)$ is equal to the distribution obtained by taking some product distribution $\mathcal{A} \times \mathcal{B}$ and conditioning it on the event $\mathsf{Good}(M, I)$. [10]

Lemma 7 (Product characterization). *For every M, I denoting Eve's information up to just before the i^{th} query, if $\Pr_{\mathcal{EXEC}(M,I)}[\mathsf{Good}(M, I)] > 0$ there exist a distribution $\mathcal{A}$ (resp. $\mathcal{B}$) over Alice's (resp. Bob's) view up to that point such that the distribution $\mathcal{GEXEC}(M, I)$ is the same as the product distribution $(\mathcal{A}\times\mathcal{B})$ conditioned on the event $\mathsf{Good}(M, I)$: $\mathcal{GEXEC}(M, I) = (\mathcal{A} \times \mathcal{B}) \mid \mathsf{Good}(M, I)$.*

Proof. We will show that for every pair of Alice/Bob views (A, B) in the probability space $\mathcal{EXEC}(M, I)$ that satisfy the event $\mathsf{Good}(M, I)$, $\Pr_{\mathcal{GEXEC}(M,I)}[(A, B)] = c(M, I)\alpha_A\alpha_B$ where α_A depends only on A, α_B depends only on B and $c(M, I)$ depends only on M, I. This means that if we let $\mathcal{A}$ be the distribution such that $\Pr_{\mathcal{A}}[A]$ is proportional to α_A, and $\mathcal{B}$ be the distribution such that $\Pr_{\mathcal{B}}[B]$ is proportional to α_B, then $\mathcal{GEXEC}(M, I)$ is proportional (and hence equal to) the distribution $\mathcal{A} \times \mathcal{B} \mid \mathsf{Good}(M, I)$.

Because $(A, B) \in \mathrm{SUPP}(\mathcal{GEXEC}(M, I))$, if (A, B) happens, it makes the event $\mathsf{Good}(M, I)$ hold, and so we have

$$\begin{aligned}\Pr_{\mathcal{EXEC}(M,I)}[(A, B)] &= \Pr_{\mathcal{EXEC}(M,I)}[(A, B) \wedge \mathsf{Good}(M, I)] \\ &= \Pr_{\mathcal{EXEC}(M,I)}[\mathsf{Good}(M, I)] \Pr_{\mathcal{GEXEC}(M,I)}[(A, B)] \ .\end{aligned}$$

On the other hand, by definition we have $\Pr_{\mathcal{EXEC}(M,I)}[(A, B)] = \frac{\Pr_{\mathcal{EXEC}}[(A,B,M,I)]}{\Pr_{\mathcal{EXEC}}[(M,I)]}$, therefore it holds that $\Pr_{\mathcal{GEXEC}(M,I)}[(A, B)] = \frac{\Pr_{\mathcal{EXEC}}[(A,B,M,I)]}{\Pr_{\mathcal{EXEC}}[(M,I)] \Pr_{\mathcal{EXEC}(M,I)}[\mathsf{Good}(M,I)]}$. The denominator of the righthand side is only dependent on M and I. The numerator is equal to $2^{-|r_A|}2^{-|r_B|}2^{-\ell|Q(A)\cup Q(B)\cup Q(I)|}$. The reason is that the necessary and sufficient condition that $(A = (r_A, H_A), B = (r_B, H_B), M, I)$ happens is that when we choose an execution (r'_A, r'_B, H') then $r'_A = r_A$, $r'_B = r_B$ and H is consistent on the queries in $Q(A) \cup Q(B) \cup Q(I)$ with the answers

[10] A similar observation was made by [8], see Lemma 6.5 there.

specified in H_A, H_B, I. Note that this will ensure that Alice and Bob will indeed produce the transcript M. Let $\alpha_A = 2^{-|r_A|}2^{-\ell|Q(A)\setminus Q(I)|}$ and $\beta_B = 2^{-|r_B|}2^{-\ell|Q(B)\setminus Q(I)|}$. Since $(Q(A) \setminus Q(I)) \cap (Q(B) \setminus Q(I)) = \emptyset$, the numerator is equal to $2^{-|r_A|}2^{-|r_B|}2^{-\ell|Q(A)\cup Q(B)\cup Q(I)|} = \alpha_A\beta_B 2^{-\ell|Q(I)|}$. Thus indeed $\Pr_{\mathcal{GEXEC}(M,I)}[(A,B)] = c(M,I)\alpha_A\beta_B$ where $c(M,I)$ only depends on (M,I).

Graph Characterization. This product characterization implies that we can think of $\mathcal{GEXEC}$ as a distribution over random edges of some bipartite graph G. Using some insights on the way this graph is defined, and the definition of our attacking algorithm, we will show that every vertex in G is connected to most of the vertices on the other side. We then show that this implies that Bob's chance of asking a query outside of I that was asked before by Alice is bounded by $O(\epsilon/n)$.

More precisely, fixing M, I that contain Eve's view up to just before the i^{th} round, define a bipartite graph $G = (V_L, V_R, E)$ as follows. Every node $u \in V_L$ will have a corresponding view A_u of Alice that is in the support of the distribution $\mathcal{A}$ obtained from Lemma 7; we let the number of nodes corresponding to a view A be proportional to $\Pr_{\mathcal{A}}[A]$, meaning that $\mathcal{A}$ corresponds to the uniform distribution over the left-side vertices V_L. Similarly, every node $v \in V_R$ will have a corresponding view of Bob B_v such that $\mathcal{B}$ corresponds to the uniform distribution over V_R. We define $Q_u = Q(A_u) \setminus Q(I)$ for $u \in V_L$ to be the set of queries *outside of* I that were asked by Alice in the view A_u, and define $Q_v = Q(B_u) \setminus Q(I)$ similarly. We put an edge in the graph between u and v (denoted by $u \sim v$) if and only if $Q_u \cap Q_v = \emptyset$. Lemma 7 implies that the distribution $\mathcal{GEXEC}(M,I)$ is equal to the distribution obtained by letting (u,v) be a random edge of the graph G and choosing (A_u, B_v).

It turns out that this graph is *dense* (i.e., every vertex is connected to almost all other vertices in the other side). The proof has two steps. The first one is to show that such graphs are "highly connected" in the sense that removing any vertex v and its neighbors from the graph, remains a small fraction of the edges in the graph. The reason is that otherwise, there is a member of Q_v which is heavy and Eve should have asked that query. The second step is to show that this notion of connectivity would imply that the graph dense (whenever the graph is bipartite). More formally, we prove the following lemma:

Lemma 8. *Let $G = (V_L, V_R, E)$ be the graph above. Then for every $u \in V_L$, $d(u) \geq |V_R|(1-2\epsilon)$ and for every $v \in V_R$, $d(v) \geq |V_L|(1-2\epsilon)$ where $d(w)$ is the degree of the vertex w.*

Proof. We first show that for every $w \in V_L$, $\sum_{v\in V_R, w\not\sim v} d(v) \leq \epsilon|E|$. The reason is that the probability of vertex v being chosen when we choose a random edge is $\frac{d(v)}{|E|}$ and if $\sum_{v\in V_R, w\not\sim v} \frac{d(v)}{|E|} > \epsilon$, it means that $\Pr_{(u,v)\in_{\mathrm{R}} E}[Q_w \cap Q_v \neq \emptyset] \geq \epsilon$. Hence because $|Q_w| \leq n$, by the pigeonhole principle there exists $q \in Q_w$ such that $\Pr_{(u,v)\in_{\mathrm{R}} E}[q \in Q_v] \geq \epsilon/n$. But this is a contradiction, because then q should be in I by the definition of the attack and hence cannot be in Q_w. The same argument shows that for every $w \in V_R$, $\sum_{u\in V_L, u\not\sim w} d(u) \leq \epsilon|E|$. Thus for every vertex $w \in V_L \cup V_R$, $|E^{\not\sim}(w)| \le \epsilon|E|$ where $E^{\not\sim}(w)$ denotes the set of edges that are not adjacent to any neighbor of w (i.e., $E^{\not\sim}(w) = \{(u, v) \in E \mid u \not\sim w \wedge w \not\sim v\}$). Now the following claim proves the lemma.

Claim. Let $G = (V_L, V_R, E)$ be a nonempty bipartite graph such that for every vertex w, $|E^{\not\sim}(w)| \le \epsilon|E|$ for $\epsilon \le 1/2$, then for all $u \in V_L, d(u) \ge |V_R|(1 - 2\epsilon)$ and for every $v \in V_R$, $d(v) \ge |V_L|(1 - 2\epsilon)$.

Proof. Let $d_L = \min\{d(u) \mid u \in V_L\}$ and $d_R = \min\{d(v) \mid v \in V_R\}$. By switching the left and right sides if necessary, we may assume without loss of generality that **(*)**: $\frac{d_L}{|V_R|} \le \frac{d_R}{|V_L|}$. Thus it suffices to prove that $1 - 2\epsilon \le \frac{d_L}{|V_R|}$. Suppose $1-2\epsilon > \frac{d_L}{|V_R|}$, and let $u \in V_L$ be the vertex that $d(u) = d_L < (1-2\epsilon)|V_R|$. Because for all $v \in V_R$ we have $d(v) \le |V_L|$, thus using **(*)** we see that $|E^{\sim}(u)| \le d_L|V_L| \le d_R|V_R|$ where $E^{\sim}(u) = E \setminus E^{\not\sim}(u)$. On the other hand since we assumed that $d(u) < (1-2\epsilon)|V_R|$, there are more than $2\epsilon|V_R|d_R$ edges in $E^{\not\sim}(u)$, meaning that $|E^{\sim}(u)| < |E^{\not\sim}(u)|/(2\epsilon)$. But this implies

$$|E^{\not\sim}(u)| \le \epsilon|E| = \epsilon\left(|E^{\not\sim}(u)| + |E^{\sim}(u)|\right) < \epsilon|E^{\not\sim}(u)| + |E^{\not\sim}(u)|/2 \ ,$$

which is a contradiction for $\epsilon < 1/2$ because the graph G is nonempty.

Proof of Lemma 6 from Lemmas 7 and 8. Let B, M, I be as in Lemma 6 and q be Bob's query which is fixed now. By Lemma 7, the distribution $\mathcal{GEXEC}(M, I)$ conditioned on getting B as Bob's view is the same as $(\mathcal{A} \times \mathcal{B})$ conditioned on $\mathsf{Good}(M, I) \wedge (\mathcal{B} = B)$. By the definition of the bipartite graph $G = (V_L, V_R, E)$ it is the same as choosing a random edge $(u, v) \in_{\mathrm{R}} E$ conditioned on $B_v = B$ and choosing (A_u, B_v). We prove Lemma 6 even conditioned on fixing v such that $B_v = B$. Now the distribution on Alice's view is the same as choosing $u \in_{\mathrm{R}} N(v)$ to be a random neighbor of v and choosing A_u. Let $S = \{u \in V_L \mid q \in A_u\}$. Then it holds that

$$\Pr_{u \in_{\mathrm{R}} N(v)}[q \in A_u] \le \frac{|S|}{d(v)} \le \frac{|S|}{(1-2\epsilon)|V_L|} \le \frac{|S||V_R|}{(1-2\epsilon)|E|} \le \frac{\sum_{u \in S} d(u)}{(1-2\epsilon)^2|E|} \le \frac{\epsilon}{(1-2\epsilon)^2 n} < \frac{3\epsilon}{2n} \ .$$

The second and fourth inequalities are because of Lemma 8. The third one is because $|E| \le |V_L||V_R|$. The fifth one is because of the definition of the attack which asks ϵ/n heavy queries, and the sixth one is because $\epsilon = \delta/13 < 1/13$. □

4.2 Efficiency of Attack: Proof of Lemma 5

The proof of attack's efficiency (i.e. Lemma 5) crucially uses the fact that the attack is *successful*, and uses the following lemma from [8].

Lemma 9 (Lemma 6.4 of [8]). *Let $Z_1, \ldots, Z_i, \ldots$ be any sequence of random variables determined by a finite underlying random variable X, let F be any event for random variable X, and let $0 \le p \le 1$. Let B_j be the event that $\Pr_X[F(X) \mid Z_1, \ldots, Z_j] \ge p$, and let $B = \bigvee_j B_j$. Then it holds that $\Pr_X[F[X] \mid B] \ge p$.*

We say that a member of the probability space $X \in \mathcal{EXEC}$ is in the event Bad_j, if at the moment that Eve is go to ask her j^{th} query from the oracle, we have $\Pr_{\mathcal{EXEC}(M,I)}[\neg\mathsf{Good}(M,I)] > 1/2$, where (M,I) are the sequence of messages and Eve's set of query/answer pairs at that moment. Let $\mathsf{Bad} = \bigvee_j \mathsf{Bad}_j$. We define the probability space $\mathcal{EXEC}(\overline{\mathsf{Bad}})$ to be the same as $\mathcal{EXEC}$ with the difference that Eve stops asking more queries whenever Bad_j happens for some j.

The proof of efficiency consists of the following two steps.

Step 1. We first use the success property of the attack (i.e., $\Pr_{\mathcal{EXEC}}[\mathsf{Fail}] \le 3\epsilon/n$) to show that $\Pr_{\mathcal{EXEC}}[\mathsf{Bad}] \le 6\epsilon$ (Lemma 10 below) which also means that $\Pr_{\mathcal{EXEC}(\overline{\mathsf{Bad}})}[\mathsf{Bad}] = \Pr_{\mathcal{EXEC}}[\mathsf{Bad}] \le 6\epsilon$. Note that $\neg\mathsf{Good}(M,I)$ implies that Fail_i has already happened for some i, and so $\neg\mathsf{Good}(M,I)$ implies Fail.

Step 2. We then show that in $\mathcal{EXEC}(\overline{\mathsf{Bad}})$ on average Eve will not ask more than $N = \frac{4n^2}{\epsilon}$ number of queries (see Lemma 11 below). Since Long is the event that Eve asks more than $\frac{n^2}{\epsilon^2} = \frac{N}{4\epsilon}$ queries, by Markov inequality we have $\Pr_{\mathcal{EXEC}(\overline{\mathsf{Bad}})}[\mathsf{Long}] \le 4\epsilon$, and therefore we will have

$$\begin{aligned}\Pr_{\mathcal{EXEC}}[\mathsf{Long}] \le \Pr_{\mathcal{EXEC}}[\mathsf{Long} \vee \mathsf{Bad}] &= \Pr_{\mathcal{EXEC}(\overline{\mathsf{Bad}})}[\mathsf{Long} \vee \mathsf{Bad}] \\ &\le \Pr_{\mathcal{EXEC}(\overline{\mathsf{Bad}})}[\mathsf{Long}] + \Pr_{\mathcal{EXEC}(\overline{\mathsf{Bad}})}[\mathsf{Bad}] \le 10\epsilon \ .\end{aligned}$$

Now we prove the needed lemmas.

Lemma 10. $\Pr_{\mathcal{EXEC}}[\mathsf{Bad}] \le 6\epsilon$.

Proof. We use Lemma 9 as follows. Let the underlying random variable be $X = \mathcal{EXEC}$, and the event $F = \mathsf{Fail}$. Let the random variable Z_j be the information that Eve learns about X after asking her $(j-1)^{\text{th}}$ query, before she asks her j^{th} query. Namely $(Z_1, \dots, Z_j)$ is equal to (M,I) of the moment she wants to ask her j^{th} query. Let $p = 1/2$, which means B_j is the event that $\Pr[\mathsf{Fail} \mid Z_1, \dots, Z_j] \ge 1/2$. Lemma 9 implies that $\Pr[\mathsf{Fail} \mid B] \ge 1/2$.

Note that $\neg\mathsf{Good}(M,I)$ at any moment implies that Fail has already happened, so Bad_j implies B_j and therefore Bad implies B. Now if $\Pr_{\mathcal{EXEC}}[\mathsf{Bad}] \ge 6\epsilon$, we would have $\Pr[\mathsf{Fail}] \ge \Pr[B \wedge \mathsf{Fail}] = \Pr[B]\Pr[\mathsf{Fail} \mid B] \ge \Pr[\mathsf{Bad}](\frac{1}{2}) \ge 3\epsilon$ which contradicts Lemma 4.

Lemma 11. *Let $\gamma = \frac{\epsilon}{2n}$, and $X = \mathcal{EXEC}(\overline{\mathsf{Bad}})$. If I denotes the set of query/answer pairs that Eve learns by the end of protocol in X, then $\mathsf{E}_X[|I|] \le \frac{2n}{\gamma} = \frac{4n^2}{\epsilon}$.*

Proof. For a fixed query $q \in \{0,1\}^\ell$, let E_q (resp. F_q) be the event (over X) that Eve (resp. Alice or Bob) asks q. By linearity of expectation we have $\mathsf{E}[|I|] = \sum_q \Pr[E_q]$ and $\sum_q \Pr[F_q] \le 2n$. We claim that $\Pr[E_q]\gamma \le \Pr[F_q]$ which would imply the lemma $\mathsf{E}[|I|] = \sum_q \Pr[E_q] \le \frac{1}{\gamma}\sum_q \Pr[F_q] \le \frac{2n}{\gamma}$.

To prove $\Pr[E_q]\gamma \leq \Pr[F_q]$, we use Lemma 9 as follows. The underlying random variable $X = \mathcal{EXEC}(\overline{\mathsf{Bad}})$ (as here), the event $F = F_q$, and the random variable Z_j is as defined in the proof of Lemma 10. Let $p = \gamma$ which means B_j is the event that $\Pr[F_q \mid Z_1, \dots, Z_j] \geq \gamma$. Lemma 9 implies that $\Pr[F_q \mid B] \geq \gamma$.

Note that if Eve asks q from the oracle when she knows $(M, I) = Z_1, \dots, Z_j$ about X, q has at least ϵ/n probability to be asked by Alice or Bob conditioned on $\mathsf{Good}(M, I)$. But $\Pr[\mathsf{Good}(M, I)] \geq 1/2$ holds in X whenever Eve wants to ask a query, and it means that q is asked by Alice or Bob with probability at least $\frac{\epsilon}{2n} = \gamma$ before. In other words when Eve asks q it holds that $\Pr[F_q \mid Z_1, \dots, Z_j] \geq \gamma$ which means that the event E_q implies B.

Therefore it holds that $\Pr[F_q] \geq \Pr[F_q \wedge B] = \Pr[B]\Pr[F_q \mid B] \geq \Pr[E_q]\gamma$.

5 Completing the Proof

To complete the proof, we need to **(a)** show how to handle protocols that are not necessarily in normal form and **(b)** show how Eve can recover the secret once she knows the intersection queries. Task **(b)** was already achieved by [8, Theorem 6.2] (although it can be shown that our attack does not need to ask any more queries to find the secret). [8] also showed how one can achieve task **(a)** using a general "compiler" that transforms general protocols to normal form. However that transformation has a quadratic blowup in efficiency that we cannot afford. We now sketch how our attack can be extended to handle general protocols without incurring this cost. (See the full version for the remaining details.)

In order to get an attack of the same $(\frac{13n}{\delta})^2$ complexity finding all the intersection queries of Alice and Bob for general form of protocols we do the following.

Attack for Seminormal Protocol. We first extend the result with the same complexity of $(\frac{13n}{\delta})^2$ queries for the attack to the "seminormal" protocols by a bit more careful analysis of the same attack given above. A seminormal protocol is a protocol in which Alice and Bob can ask either zero or one query in each of their rounds. Again Alice and Bob ask at most n queries each, but the number of rounds can be arbitrary larger than n.

Roughly speaking, the reason that the same attack as above works for seminormal protocols is that although there are $\Omega(n^2)$ number of rounds in the new seminormal protocol, we only need to bound the probability that Eve misses an intersection query for the first time whenever Alice or Bob does ask a query in their turn (and there are only $2n$ such queries). Assuming that it is Bob's turn, if we fix the query he asks we can still bound the probability that the query is the first missing intersection query using Lemma 6. That is because in Lemma 6 the statement holds even conditioned on Bob's computation (and his query in particular) being fixed.

Compiling into Seminormal Form. Any protocol can be simply changed into a seminormal protocol without increasing n or loosing the security. To see why, suppose it is Bob's turn and he is going to ask $k \leq n$ queries from the oracle before sending a message to Alice. Alice and Bob can blow this single round into $2n - 1$ number of rounds in which Alice does nothing other than sending $\perp$ to Bob and it lets Bob to ask his queries in different rounds. He sends the actual message in the last round among the $2n - 1$ new rounds. The total number of rounds will be $O(n^2)$, but the number of queries that Alice and Bob together ask will still remain $2n$ as before.

Acknowledgements. We thank Russell Impagliazzo for useful discussions, and also for his warning that attempting to prove an $O(n^2)$ bound for this problem leads naturally to conjecturing (and even conjecturing that you proved) intermediate results that are simply not true. He was much more prescient than we realized at the time.

References

1. Merkle, R.: Secure communications over insecure channels. Communications of the ACM 21(4), 294–299 (1978)
2. Cachin, M.: Unconditional security against memory-bounded adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 292–306. Springer, Heidelberg (1997)
3. Biham, E., Goren, Y.J., Ishai, Y.: Basing weak public-key cryptography on strong one-way functions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 55–72. Springer, Heidelberg (2008)
4. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Proc. 30th STOC, pp. 209–218. ACM, New York (1998)
5. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
6. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the First Annual Conference on Computer and Communications Security, November 1993, pp. 62–73. ACM, New York (1993)
8. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proc. 21st STOC, pp. 44–61. ACM, New York (1989); Full version available from Russell Impagliazzo's home page
9. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Annual Symposium on Theory of Computing, May 22–24, pp. 212–219 (1996)
10. Bennett, C.H., Brassard, G., Ekert, A.: Quantum cryptography. SCIAM: Scientific American 267 (1992)
11. Brassard, G., Salvail, L.: Quantum merkle puzzles. In: International Conference on Quantum, Nano and Micro Technologies (ICQNM), pp. 76–79. IEEE Computer Society, Los Alamitos (2008)
12. Barak, B., Mahmoody-Ghidary, M.: Merkle Puzzles are Optimal. Arxiv preprint arXiv:0801.3669v1 (2008); Preliminary version of this paper. Version 1 contained a bug that is fixed in this version
13. Sotakova, M.: Breaking one-round key-agreement protocols in the random oracle model. Cryptology ePrint Archive, Report 2008/053 (2008), `http://eprint.iacr.org/`

Position Based Cryptography*

Nishanth Chandran[1], Vipul Goyal[1,**],
Ryan Moriarty[1], and Rafail Ostrovsky[2,***]

[1] Department of Computer Science, UCLA
{nishanth,vipul,ryan}@cs.ucla.edu
[2] Department of Computer Science and Mathematics, UCLA
rafail@cs.ucla.edu

Abstract. We consider what constitutes *identities* in cryptography. Typical examples include your name and your social-security number, or your fingerprint/iris-scan, or your address, or your (non-revoked) public-key coming from some trusted public-key infrastructure. In many situations, however, **where you are** defines your identity. For example, we know the role of a bank-teller behind a bullet-proof bank window not because she shows us her credentials but by merely knowing her location. In this paper, we initiate the study of cryptographic protocols where the identity (or other credentials and inputs) of a party are derived from its *geographic location.*

We start by considering the central task in this setting, i.e., securely verifying the position of a device. Despite much work in this area, we show that in the Vanilla (or standard) model, the above task (i.e., of secure positioning) is impossible to achieve. In light of the above impossibility result, we then turn to the Bounded Storage Model and formalize and construct information theoretically secure protocols for two fundamental tasks:

– Secure Positioning; and
– Position Based Key Exchange.

We then show that these tasks are in fact *universal* in this setting – we show how we can use them to realize Secure Multi-Party Computation.
Our main contribution in this paper is threefold: to place the problem of secure positioning on a sound theoretical footing; to prove a strong impossibility result that simultaneously shows the insecurity of previous attempts at the problem; and to present positive results by showing that the bounded-storage framework is, in fact, one of the "right" frameworks (there may be others) to study the foundations of position-based cryptography.

* Full version available on eprint. Research supported in part by NSF grants 0716835, 0716389, 0830803.

** Research supported in part by above NSF grants and a Microsoft Graduate Fellowship.

*** Research supported in part by an IBM Faculty Award, Xerox Innovation Group Award, NSF grants 0430254, 0716835, 0716389, 0830803 and U.C. MICRO grant.

1 Introduction

1.1 Motivation

In cryptography, typically a party will possess a set of credentials determining: its identity, what tasks it can do, which protocols it can participate in and so on. These set of credentials will typically correspond to the party having some of the following attributes: some secret information (e.g., a secret key), authenticated information (e.g., a digitally signed certificate from a trusted entity), biometric feature and so on. In this paper, we ask the following question: can the *geographical position* of a party be one of the credentials? The geographical position of a party is valuable in a number of natural settings. We give a few examples:

– **Position based Secret Communication.** Consider communication between different military establishments. For example, the Pentagon in Washington D.C. might want to send a message (having some classified information) such that it can only be read by an individual present at the US military base in South Korea. In a traditional solution, the South Korean military base will have a secret key to decrypt the message. However, the enemy might try to break into the military base computers to capture this key. It would be desirable to add an additional layer of security that would guarantee that anyone reading the message is *physically* present at the South Korean base.
– **Position based Authentication/Signatures.** In the above example, suppose the South Korean military base wants to send some information to the Pentagon. It would be desirable for the Pentagon to have a guarantee that the message was indeed sent from the geographical position of the military base.

Indeed, the above list is not exhaustive. One could think about *position based access control* (where access to a resource needs to be restricted to certain locations, e.g., a printer or fax machine is accessible only to people inside some set of offices) and *pizza delivery* (where the pizza company first wants to verify that the person placing the order is indeed located at the delivery address he specified). To perform such "position specific" tasks, we introduce the notion of *position based cryptography.*

The first natural question that arises is: "Can you convince others about where you are?". More precisely, we have a prover who claims be at a geographical position P. There is a set of remote verifiers (or in other words, a positioning infrastructure) who wish to make sure that the prover is indeed at position P as claimed (for example, by executing a protocol with that prover). We call the above problem as "Secure Positioning". The question of secure positioning is a fundamental one and deals with designing a system which enables a prover to communicate back and forth with a group of verifiers to give them an *interactive proof of its geographic position.*

The problem of secure positioning is well studied in the security community (see e.g., [3,5,6,20,21]). The de-facto method to perform secure positioning is based on the time of response technique where the messages travel with the speed of radio waves which is equal to the speed of light (this is similar in nature to how the commercial GPS systems work, see section 1.3). At a high level, the verifiers will send messages to the device and will measure the time taken to receive a response. Although there have been several proposed protocols for secure positioning, all of them are completely insecure under the so called "collusion attack". That is, if a set of (possibly cloned) provers collude together and work in a controlled manner during the protocol execution, the provers will be able to convince the verifiers that the verifiers are talking to a prover at position P (even though none of the adversarial provers may be at P). We in fact show that, unfortunately, such an attack is unavoidable. That is, it is impossible to have secure protocols for positioning in this Vanilla model (even if one is willing to make computational assumptions). Hence, we cannot hope to realize most of the meaningful position based tasks.

In light of the above drawbacks, in this paper we explore the intriguing possibility if secure positioning protocols exist which can resist collusion attacks. In search of an answer to this question, we turn to the bounded storage model (BSM), introduced by Maurer [18]. Quite surprisingly, this model turns out to be a right model for proving the security of position-based cryptographic tasks. We first construct a protocol for information theoretic secure positioning in this model. To our knowledge, this is the first protocol which is secure even against collusion attacks. Although secure positioning is an important step, the full power of position based cryptography can only be realized if we achieve key exchange with the device at a particular geographic position. Hence we introduce position based key exchange and present two protocols to achieve it in the BSM. Our first protocol achieves security against a computationally bounded adversary (in the BSM). In this protocol, we achieve key exchange between the verifiers and any device at position P that is enclosed within the tetrahedron formed between 4 verifiers in 3-dimensional space. Our second protocol achieves information theoretic key exchange between the verifiers and devices at positions P that lie in a specific geometric region (characterized by a condition that P must satisfy) within the tetrahedron.

Note that we are interested only in verifying the position claim of devices that are within the tetrahedron enclosed between the 4 verifiers. This is not a limitation, since apriori, we are restricting, by geographical bounds, the locations where an honest device can be located (such as inside a room, to get access to a printer or a hard drive). If a device makes a position claim that lies outside of this region, we reject the claim without any verification. We stress, however, that we do not make *any* assumption about the positions of adversaries in the system. In particular, this freedom for the adversarial devices guarantees that no set of adversaries (some of whom may even be outside of the tetrahedron) can falsely prove that any one of them is at position P inside the tetrahedron as long as none of them are at position P.

1.2 The Two Models Considered

The Vanilla Model. We now informally describe the Vanilla model. We have a device (also referred to as the prover) who is located at a position P (where P is a point in a d-dimensional Euclidean space). There exists a set of verifiers $\{V_1, V_2, ..., V_m\}$ at different points in the d-dimensional space, such that P lies inside the tetrahedron enclosed by the verifiers. The verifiers are allowed to execute a protocol with the prover to achieve some task. More precisely, a verifier can send messages to the prover at different points in time (with a speed up to the speed of radio waves) and also record the messages which are received from it (along with the time when they are received). The verifiers have a secret channel among themselves using which they can coordinate their actions by communicating before, during or after protocol execution. There could be multiple adversaries with possibly cloned devices who share a covert channel and collude together. This setting is referred to as the Vanilla model.

The Bounded Storage Model. The bounded storage model (BSM) was introduced by Maurer in [18] and has been the subject of much work [18,4,1,9,17,22,19,13,10]. Very roughly, this model assumes that there is a bound on the amount of information that parties (including an adversary) can store. It assumes the existence of random strings, having high min-entropy, available to the parties at the beginning of the protocol. An adversary is allowed to retrieve and store an arbitrary function of this random string, as long as the length of the output of the function is not longer than the adversary's storage bound. We assume that parties can broadcast random strings having high min-entropy, but cannot store these strings. A closely related model to the BSM is the bounded retrieval model (BRM), introduced and studied in various related contexts by Di Crescenzo et al [8] and Dziembowski [11,12]. This model assumes that parties *can store* information having high min entropy, but an adversary can only retrieve part of it. Recently, Dziembowski and Pietrzak [14] introduced *intrusion resilient secret sharing* where shares of a secret (stored on different machines) are made artificially large so that it is hard for an adversary to retrieve a share completely, even if it breaks into the storage machine. We note that in the current work, we use the work of [14] on Intrusion Resilient Secret Sharing schemes as a starting point. We build and extend these techniques by combining them with geometric arguments to prove the security of our protocol.

In the context of position based cryptography, by bounded storage model, we mean the Vanilla model setting where the verifiers can broadcast information having high entropy (or control a randomness source which can) such that the adversaries can only retrieve and store, say, a constant fraction of this information as it passes by at high speed. The assumption that the adversaries cannot retrieve (and store) all the information that goes by seems plausible in our setting since the information travels at a very high speed (particularly when, e.g., the verifiers have several sources broadcasting information at different frequencies). The reason we call our model bounded storage (as opposed to bounded retrieval) is that we do not assume that verifiers (and the honest prover) can fully store the broadcasted information themselves.

1.3 Related Work

Secure Positioning. We remark that the problem of position-based cryptography as such has not been studied before. However, secure positioning is a well-studied problem in the field of wireless security. There have been several proposed protocols ([2,20,23,3,6,21,24]). All these protocols are susceptible to the collusion attack outlined earlier. One can get around this problem of multiple cloned adversaries by assuming a setup phase where the verifiers give an *unclonable* tamper-proof hardware [16,15](having some secret information) to all possible future provers. However in the current work, we focus on the setting where the *only* credential needed by a prover is its geographical position.

In [5], a model is considered, that makes the assumption that there can exist verifiers that are covert or hidden to provers and adversaries. Based on this, they provide solutions to secure positioning. The protocols in [5] are also susceptible to multiple colluding adversaries, although the attack required is more subtle than in other cases. We describe this attack, as well as give a detailed description of related work on secure positioning and the BSM in the full version of this paper [7].

Global Positioning System. The problem addressed by the global positioning system (GPS) is complementary to the one considered in our work. In GPS, there is device trying to determine *its own* geographic position with the aid of various satellites (in a non-adversarial setting). The GPS satellites continually broadcast information in a synchronized manner with the speed of light. The time taken by the information broadcast by various satellites to reach a GPS receiver enables the receiver to compute its position using triangulation techniques.

1.4 Our Contributions

In this paper, we give the following results towards developing a theory of position based cryptography:

– We begin with a lower bound for the Vanilla model in Section 3. We show that there does not exist a protocol in the Vanilla model using which a group of verifiers can securely verify the location claim of a prover. The impossibility is obtained via an explicit attack which does not depend on the computational power of the parties. To begin with, the lower bound holds if all the parties (i.e., the verifiers, the honest prover and the adversaries) are given unbounded computational power. Further, it holds even if the verifiers are given unbounded computational power but the adversaries (and thus obviously the honest prover) are restricted to being probabilistic polynomial time (PPT) machines (i.e., one may make cryptographic hardness assumptions). Finally, the protocols in [5] additionally assume the existence of hidden and mobile base stations and present protocols for secure positioning. In the full version of this paper [7], we describe explicit attacks breaking the security of these protocols in common settings (where a prover learns the response of the success or failure of the protocol). With the impossibility of this most fundamental task, we cannot hope to perform most other meaningful position based tasks (including position based key exchange) in the Vanilla model.

– Given the above severe lower bound, the task of now choosing a model in which protocols for secure positioning exist becomes a tricky one. One of the main technical contributions of this paper is to connect the bounded storage model to position based cryptography. Remarkably, bringing these seemingly unrelated ideas together enables us to achieve meaningful and elegant protocols and proofs of security for position based cryptography.
– In the BSM, we give a protocol for secure positioning (in Section 5) which is provably secure against any number of (possibly computationally unbounded) adversaries colluding together, as long as the total amount of information they can retrieve and store is bounded. To our knowledge, this is the first protocol for positioning which does not fail against collusion attacks. We also describe, in Section 6, how our protocol for secure positioning can be compiled with any unauthenticated computationally secure key exchange protocol (like Diffie-Hellman) to achieve computationally secure position based key exchange in the BSM. That is, only a prover who is at a designated position P will receive the key to be shared (even under the presence of any number of PPT adversaries with bounded storage).
– We then present a protocol (in Section 7) that does information theoretically secure key exchange between the verifiers and a device at P. The construction of such a protocol turns out to be surprisingly intricate. While our secure positioning (and computationally secure position based key exchange) can handle claims of all positions P that lie within the tetrahedron formed between 4 verifiers in 3-dimensional space, our information theoretic key exchange protocol can handle positions P that lie in a specific region (which we characterize, using a geometric argument, by a condition that P must satisfy) within the tetrahedron. In the full version, we show (for a few example cases) that this region is a large fraction of the enclosed tetrahedron and also provide some figures containing what this region looks like (for various placements of the 4 verifiers). In order to show the security of our protocol, we need to prove delicate timing arguments (based on geometric properties) as well as prove that the protocol of [14] is secure even in the case when multiple parallel adversaries can gain access to the machines and may collude after they have finished accessing the machines.
– Using the above two fundamental protocols as building blocks, we demonstrate that the protocols for more complex tasks can be readily constructed. We consider the problem of establishing a secure channel between two devices (such that each device has a guarantee on the geographic position of the device at the other end). After establishing pairwise secure channels, a group of devices can perform "position based" multi-party computation, where associated with each input, there is a guarantee about the position of the device giving that input. We also discuss the setup of a position based public key infrastructure, where a certificate provides a guarantee about the position (as opposed to the identity) of the owner of the public key in question (We discuss these applications in further detail in the full version.). We remark that the

above is obviously not intended to be an exhaustive list of applications and one can construct protocols for several other tasks.

Our results do not require any pre-sharing of data (cryptographic keys and so on) between the prover and the verifiers. The only required credential of a prover is its *real geographic position.* We present the high level ideas of the model and constructions and defer the full formal proofs to the full version [7].

Open Problem: Other Models for Position Based Cryptography. By turning to the bounded storage model, we are able to provide the first provably secure constructions of cryptographic tasks that use position as an identity. Given our strong impossibility results in the Vanilla model, an important open question is: do there exist other natural models that allow us to obtain positive results of similar nature?

2 The Model

In this section, we briefly discuss our model. More details can be found in the full version. There are three types of parties in our model: Prover, Verifier and Adversary. We treat time and space as "continuous" (rather than discrete). We assume that messages travel at a speed equal to that of radio waves (which is the same as the speed of light). In the beginning, each party (prover, verifiers and adversaries) is given as input, party's own position (as a point in the d-dimensional space), the position of all verifiers and the security parameter κ. The verifiers and the adversaries are given the claimed position of the prover.

The parties can send out the following two types of messages : (a) Broadcast messages: A broadcast message originating at a position P travels in concentric hyperspheres centered at P in all directions, (b) Directional messages: A directional message, instead of traveling in all directions, travels only in a specific direction specified by a sector. Such messages can be sent using directional antennas. Additionally, verifiers have a private channel among themselves which allows them to talk to each other secretly. Adversaries also have a private (and covert) channel among themselves which allows them to talk to each other secretly such that no verifier suspects any adversarial activity. More details about these messages (along with formal definitions of secure positioning and key exchange) can be found in the full version.

The above is our so called Vanilla Model where we prove the impossibility of realizing the most basic position based task (i.e., secure positioning). We assume that parties can send directional messages in the Vanilla model in order to prove a strong lower bound. As noted earlier, all our positive results are in the BSM. Our bounded storage model is the same as the Vanilla model except for the following changes:

– Verifiers "possess" a reverse block entropy source (defined formally in the full version) capable of generating strings with high min-entropy, say $(\delta + \beta)n$, where n is the length of the string (and $0 < \delta + \beta < 1$; it is also called min-entropy rate). By possessing a reverse block entropy source, we mean that either the verifier itself is capable of generating such a stream of high min-entropy, or it has a randomness source (located at the same point in space as itself) which generates and broadcasts such a stream. We do not assume that the verifiers can retrieve and store the broadcasted stream of data themselves. Generating a lot of entropy is easy; one can think of an "explosion" which generates a lot of noise that can be measured but not stored.

– There exists a bound βn on the total amount of information the adversaries can store as the information passes at a high speed. The storage bound βn could be any constant fraction of the min-entropy $(\delta+\beta)n$. The honest parties (including the verifiers) are required to have a storage capacity of only $O(\kappa \cdot \log(n))$.
– Verifiers and provers cannot send directional messages. We however do not restrict the adversary from sending directional messages.
– Let X be a string having large min-entropy as before. The sender (which is a verifier) generates X and sends it out. Any receiver gets to retrieve and store $f(X)$ (for any arbitrary f) in a way such that the total amount of information which it has retrieved does not exceed the storage bounds. In case a party receives multiple strings simultaneously, it can retrieve information from these strings, in any order, any number of times (i.e., we do not restrict the adversaries to retrieve information from a string only once) as long as the total memory bound is not violated on the amount retrieved.

Observe that the last step above also enforces that any information about a string X (having large min-entropy) that is sent from one adversary to the other is also bounded (since an adversary gets to retrieve and resend only $f(X)$). This rules out simple "reflection attacks" to create a huge storage (where a pair of adversaries close to each other just keep reflecting the string X to each other hence essentially storing X thus violating the bounded storage assumption).

Relaxing Assumptions. For clarity of exposition during our positive results, we make the assumption that the devices can read bits from the stream and perform computations instantaneously. We refer the reader to the full version for details on how to remove this assumption.

3 Lower Bound on Secure Positioning in the Vanilla Model

We now show a lower bound for the Vanilla model. We show that there does not exist a protocol in the Vanilla model using which a group of verifiers can securely verify the location claim of a prover. The impossibility is obtained via an explicit attack which does not depend on the computational power of the parties. To begin with, the lower bound holds if all the parties (i.e., the verifiers, the honest prover and the adversaries) are given unbounded computational power. Further, it holds even if the verifiers are given unbounded computational power but the adversaries (and thus obviously the honest party) are restricted to being PPT machines (i.e., one may make cryptographic hardness assumptions). Finally, we present a few extensions of our lower bound in the full version.

Theorem 1. *There does not exist a protocol to achieve secure positioning in the Vanilla model.*

Proof. Let there be n verifiers $\{V_1, V_2, ..., V_n\}$ that take part in a protocol to verify that a prover is at a position P . We show that for any protocol, there exists a set of l adversaries (l to be defined later) who can interact with the verifiers in such a manner that it is impossible to distinguish if the verifiers are interacting with an adversary or the actual prover.

Consider the hypersphere of radius r around position P such that the distance between V_i and P for all i be strictly greater than r. In other words, we require that r is such that no verifier is present within the hypersphere of radius r centered at position P. For all i, let the straight line joining V_i and P intersect the hypersphere at position I_i. Let there exist $l \leq n$ such intersection points. We note that l could be less than n because, two (or more) verifiers $V_i, V_j, i \neq j$ may be such that P, V_i and V_j lie on the same straight line in d-dimensional space. We place adversaries $A_1, A_2, ..., A_l$ at points $I_1, I_2, ..., I_l$. The verifiers may run an interactive protocol with the prover in order to verify that the prover is at position P. We show that these l adversaries together can simulate the execution of the protocol in such a way that the verifiers cannot distinguish between an execution in which they are interacting with the prover at P and an execution in which they are interacting with these set of adversaries.

Any verification protocol is a series of messages (along with corresponding times), each being from one of the n verifiers to the prover or vice-versa. The verifiers can then verify the position of the prover by analyzing the message they sent and the response they got (along with corresponding times). We give a strategy for every A_m such that the adversaries together can prove that they are at position P.

Let the time taken for any message to travel between V_i and P be T_i. Note that the distance between A_m, for all m, and P is fixed (equal to r). Hence, let the time taken for a message to travel between A_m (for all m) and P be α. Let the set of verifiers that lie on the same straight line that connects A_m and P be $\mathcal{V}_m$. Let the distance between two adversaries A_m and $A_{m'}$ be $dist(m, m')$ (note that $dist(m, m) = 0$).

Now during the protocol execution, every A_m does the following. A_m only listens to messages sent by all $V_i \in \mathcal{V}_m$ and ignores messages sent by other verifiers. A_m is at a location such that all the messages sent by V_i (s.t., $V_i \in \mathcal{V}_m$) to the point P would be received by it (irrespective of whether V_i sends a broadcast message or a directional message). Lets say that a message M is received from a verifier V_i. For every adversary $A_{m'}$ (including itself, i.e., $1 \leq m' \leq l$), A_m internally delays M by the duration of time $delay(m, m') = 2\alpha - dist(m, m')$, and then sends it to $A_{m'}$ over the covert channel. Hence, every single adversary (including A_m itself) would receive the message at time 2α (over the covert channel) after the time when A_m receives it from V_i (over the broadcast or directional channel).

For every adversary A_m, now assume that the protocol requires the honest prover to send a reply message, at time t, in directions such that verifiers in set $\mathcal{V}_m$ would receive it (note that since all of them are in a straight line in the same direction of point P, either all of them would receive it or none would). In that case, A_m computes the response message using its view over the covert channel so far and sends it at time $t + \alpha$ using a directional message (such that only verifiers in $\mathcal{V}_m$ receive it). However, A_m does not send any messages to V_i for $V_i \notin \mathcal{V}_m$ (if the verifiers in other sets are required to receive this message as well, they will be "taken care of" by other adversaries near them).

The following simple argument shows that every adversary A_m runs exactly a copy of the execution of the prover, only at a time α later. Once this is shown, since it takes time T_i for a prover to send a response to V_i when $V_i \in \mathcal{V}_m$, and it takes A_m only time $T_i - \alpha$, the exact same response will reach V_i at exactly the same instance of time (irrespective of whether it originated at P or at the location of A_m).

We show that the following two statements are true. $delay(m, m')$ is a non-negative value for all m, m' and every message which reaches the prover at P will reach all the adversaries after exactly time α. This will prove that all adversaries run exactly the same copy of the prover, but at a later instance of time.

The first statement follows trivially from triangle inequality. For the second statement, assume that verifier V_i sends a message to the prover at time t. Let m be such that $V_i \in \mathcal{V}_m$ and t' be the time taken for a message to travel between V_i and A_m. The honest prover clearly receives the message at time $t+t'+\alpha$. The adversary A_m receives the message at time $t + t'$ and hence all the adversaries receive it at time $t + t' + 2\alpha$ over the covert channel.

This proves that all adversaries run exactly the same copy of the prover, but at an instance α later. Hence, any execution of a protocol run between the n verifiers and the prover can be simulated by l adversaries running the protocol with the n verifiers. □

We remark here that the above impossibility result holds even in a stronger model where there is a fixed bound on the number of adversaries, as long as this bound can depend on the number of verifiers in the system (but not on the secure positioning protocol itself). This motivates our search for alternative models, where we do not restrict the number of adversaries and still achieve positive results.

4 Preliminaries

Vadhan [22], introduced BSM pseudorandom generators (PRG). Informally, for string X sampled from a distribution having high min-entropy and for a uniformly random seed K, the distribution of the output of the BSM PRG (denoted by $\mathrm{PRG}(X, K)$), is statistically close to the uniform distribution of appropriate length even when given K and $A(X)$ where A is any arbitrary function with bounded output length. We introduce a relaxation of BSM PRGs, which we call BSM entropy generators (EG). The difference between a BSM EG and a BSM PRG is that the output distribution of a BSM EG is only guaranteed to have high min-entropy, and not necessarily be close to the uniform distribution. We refer the reader to the full version for formal details about the definitions, constructions and instantiations.

5 Secure Positioning in the Bounded Storage Model

In this section, we propose protocols for secure positioning in the BSM. We shall build upon the primitives described in Section 4. To make the intuition clear, we first give a secure positioning protocol for 1-dimension.

5.1 Secure Positioning in 1-Dimension

For 1-dimension, we employ two verifiers, denoted by V_1 and V_2 (which send messages with the speed of radio waves). We assume that the position P being claimed by the prover is located between V_1 and V_2. Our protocol is secure against an arbitrary number of adversaries colluding together to prove a position P, as long as the total information that these adversaries can store during the protocol is bounded. We let βn denote the aforementioned storage bound. Verifier V_1 is assumed to possess a random source $X_1, X_2, \cdots$ which is a reverse block entropy source of minimum entropy rate $\delta + \beta$, where $X_i \in \{0,1\}^n$.

We shall use a (ε, ψ)-secure BSM entropy generator EG: $\{0,1\}^n \times \{0,1\}^\ell \rightarrow \{0,1\}^m$ as discussed in the previous section. We choose the input size ℓ such that $\varepsilon + 2^{-\psi}$ is negligible in the security parameter κ. An example of a fast BSM EG, which is just a random sampler requiring no computations at all, is presented in the full verison.

Before the protocol starts, the verifier V_1 selects a key $K \xleftarrow{R} \{0,1\}^\ell$ and sends it to verifier V_2 over the private channel (using a private multicast message). Let t and t' be the time taken for radio waves to reach P from V_1 and V_2 respectively. Verifier V_1 sends out X from the reverse block entropy source such that X has min-entropy $(\delta + \beta)n$. At the same time, V_1 computes $\mathrm{EG}(X, K)$ and stores it on its output tape. Let T be the time at which X reaches P. Verifier V_2 sends the key K out at a time such that it *meets* X at time T at the position P. More precisely, X and K are sent at times $(T - t)$ and $(T - t')$ by V_1 and V_2 respectively.

At time T, the prover claiming to be at position P evaluates $y = \mathrm{EG}(X, K)$ and sends it back to the verifier V_1. Verifier V_1 verifies that the string y is received at time $(T + t)$ and that it equals $\mathrm{EG}(X, K)$. If these verifications succeed, the position claim of the prover is accepted and it is assumed to be indeed at position P. Otherwise, the position claim is rejected.

The protocol clearly satisfies the completeness property since an honest prover at position P will have both X and K available at time T and hence it can compute y (by asking the hypothetical ITM P_{env} to compute the function

EG$(.,K)$.) and report it back to V_1 by time $(T+t)$. We discuss the security below:

Theorem 2. *The 1-dimension secure positioning protocol is secure against an arbitrary number of adversaries colluding together, with the total adversary information storage bounded by βn.*

Proof. Suppose there exists an adversarial strategy with which a set of adversaries, none of which is at position P, are able to report back the correct y to the verifier V_1 at time $(T+t)$ with a non-negligible probability in the security parameter. We show that the above contradicts the properties of the EG.

We consider the state of the system at time T. X and K are at position P. Let there be g adversaries between V_1 and P and the information they have retrieved about X be $S_1, S_2, ..., S_g$ respectively. Let S denote the combined information $S_1 \cup S_2 \cup ... \cup S_g$. Clearly since K has not yet crossed P, S is an arbitrary function of X alone. Further, $|S| \leq \beta n$ since βn is the total storage bound. Now we have the following:

Lemma 1. *The string y to be sent to the verifier V_i at time $(t+T)$, can be an arbitrary function of S and K alone. More formally, given an adversarial strategy to compute y in our setting, there exists a simulator that outputs y only given S and K (and not the stream X).*

The above lemma holds because (a) S is the only information stored by the adversaries between V_1 and P, (b) there is no adversary at P, and, (c) any information about X between P and V_2 at time T cannot reach V_1 by time $(t+T)$.

Hence we have $y = A(S,K)$, where $A(.,.)$ is any arbitrary adversarial algorithm. However, given S and K, using properties of the BSM EG, the probability of an adversary correctly guessing y is upper bounded by $\varepsilon + 2^{-\psi}$. But $\varepsilon + 2^{-\psi}$ is negligible in the security parameter by our choice of ℓ. Thus we have reached a contradiction. □

5.2 Secure Positioning in 3-Dimensions

We generalize the above protocol to obtain a protocol for secure positioning in 3-dimensional space. βn is the total adversary information storage bound. We use 4 verifiers denoted by $V_1, \cdots, V_4$ possessing reverse block sources of minimum entropy $(\delta + \beta)n$ that output strings X_i. Position P being claimed by the prover is enclosed in the tetrahedron defined by these 4 verifiers. t_i is the time taken for radio waves to reach the point P from verifier V_i. PRG:$\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^m$ is an ε-secure BSM pseudorandom generator. We choose the parameters such that $\varepsilon + 2^{-m}$ is negligible in the security parameter. In order for the verifiers to themselves compute the response expected from the prover, we first assume that verifiers can store the X_i values. We later show how this assumption can be removed. The protocol is illustrated in Figure 1. For more details, refer the full version.

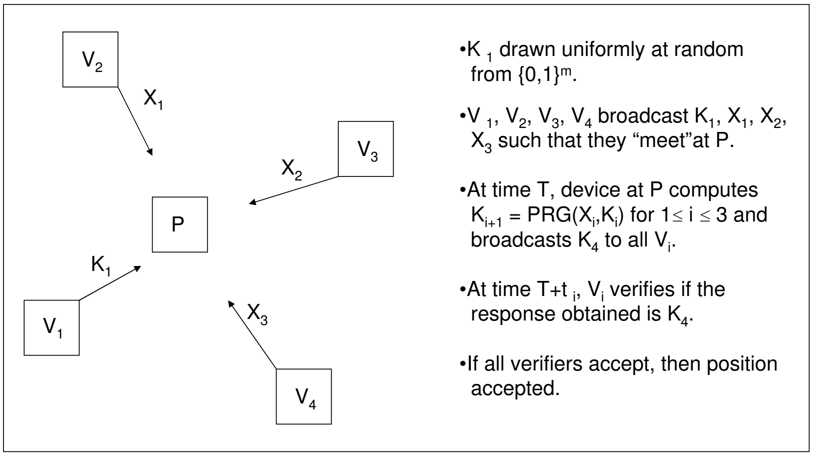

V2
X1
V3
X2
P
K1
V1
X3
V4
•K 1 drawn uniformly at random from {0,1}m.
•V 1, V2, V3, V4 broadcast K1, X1, X2, X3 such that they "meet"at P.
•At time T, device at P computes Ki+1 = PRG(Xi,Ki) for 1≤ i ≤ 3 and broadcasts K4 to all Vi.
•At time T+t i, Vi verifies if the response obtained is K4.
•If all verifiers accept, then position accepted.

Fig. 1. Secure positioning protocol in 3-Dimensions

The completeness follows from the fact that verifiers can compute K_4 from the stored X_i values and the prover can also compute K_4 since all the information required is present jointly at P at time T. The security of this protocol is proven using techniques from the proof of security of the protocol for 3-dimensional position based key exchange that is discussed in Section 7 (note that position based key exchange implies a protocol for secure positioning).

We now describe, how to remove the assumption that verifiers can store strings drawn from their respective reverse block entropy sources. Note that the problem we face when verifiers cannot store the large strings is that verifiers have no way of verifying the response of the prover. This is because, when for example, V_3 broadcasts string X_2, it does not know the key K_2 used to compute K_3 from X_2. We get around this problem as follows. The verifiers pre-determine the keys K_1, K_2, K_3, K_4 that are to be used at every iteration of the application of the PRG. Now, the expected response of the prover, K_4 is known before protocol execution to all verifiers. The protocol is as follows:

1. V_1, V_2, V_3 and V_4 pick keys $K_1, K_2, K_3, K_4 \overset{R}{\leftarrow} \{0,1\}^m$ and broadcast them over their private channel.
2. V_1 broadcasts key K_1 at time $T - t_1$. V_2 broadcasts X_1 at time $T - t_2$ and simultaneously also broadcasts $K'_2 = \text{PRG}(X_1, K_1) \oplus K_2$. Similarly, V_3 broadcasts $(X_2, K'_3 = \text{PRG}(X_2, K_2) \oplus K_3)$ at time $T - t_3$ and V_4 broadcasts $(X_3, K'_4 = \text{PRG}(X_3, K_3) \oplus K_4)$ at time $T - t_4$.
3. At time T, the prover at position P computes messages $K_{i+1} = \text{PRG}(X_i, K_i) \oplus K'_{i+1}$ for $1 \leq i \leq 3$. The prover returns K_4 to all verifiers.
4. All verifiers check that the string K_4 is received at time $(T + t_i)$ and that it equals the K_4 that they pre-picked. If these verifications succeed, the position claim of the prover is accepted and it is assumed to be indeed at position P. Otherwise, the position claim is rejected.

The completeness of this protocol is as follows. Note that since the verifiers picked K_4 before the execution of the protocol, they can verify the response of a prover without storing any of the large random strings. To informally argue security of the protocol, note that in this protocol, instead of using the output of the PRG as an input key in the next round, one treats the output as one secret share of the key to be used. The other share of this key is broadcast in the clear. Now, if one of the shares of an additive secret sharing scheme is random, then the secret is hidden. Hence, by the security of the protocol in which verifiers could store the large random strings, it follows that this protocol is also secure.

6 Computational Position Based Key Exchange

Informally, position based key exchange should have the property that if there is a prover at the claimed position P, then at the end of the protocol, the verifiers should share a uniform key K with it while for a group of colluding adversaries (none of whom is at P) K should look indistinguishable from a key drawn uniformly at random. This also implies that in the absence of a prover at position P, such a group of adversaries should be unable to execute the key exchange protocol on their own to obtain a shared key with the verifiers. In the full version [7], we show how to compile any 1-round *information theoretically* secure positioning protocol SP in our bounded storage model along with any unauthenticated key-exchange protocol KE to obtain an authenticated computational position based key exchange protocol CKE in the BSM.

7 Information Theoretic Position Based Key-Exchange

In this section, we present an information theoretic protocol to achieve position based key exchange. The overview of our protocol can be found in Figure 2. We start with some intuition behind our protocol and the techniques required to prove its security. Let us first consider the case of one dimension. We extend the protocol for secure positioning in one dimension presented earlier for the case of key exchange as follows. Instead of only one verifier V_2 sending a "large" string (drawn from a reverse block entropy source), both the verifiers send one large string each. More precisely, the verifier V_1 sends a key K_1 and a large string X_2 while the verifier V_2 sends a large string X_1 such that all of them meet at the claimed position P at the same instance of time T. The computation of the final key K_3 is done by the prover as follows: set $K_2 = \text{PRG}(X_1, K_1)$, $K_3 = \text{PRG}(X_2, K_2)$.

To see the intuition behind why this protocol is a secure one dimensional information theoretic position based key exchange, let us consider the state of the system at time T. Adversaries between V_1 and P (say, adversaries of type I) have stored $(K_1, A(X_2, K_1))$ while adversaries between P and V_2 (say, adversaries of type II) have stored $A(X_1)$. After time T, the adversaries of type I can compute K_2 thus transitioning their state to $(K_2, A(X_2, K_1))$ while adversaries of type II can only transition their state to $A(X_1), K_1, A(X_2, K_1)$. Thus it seems that

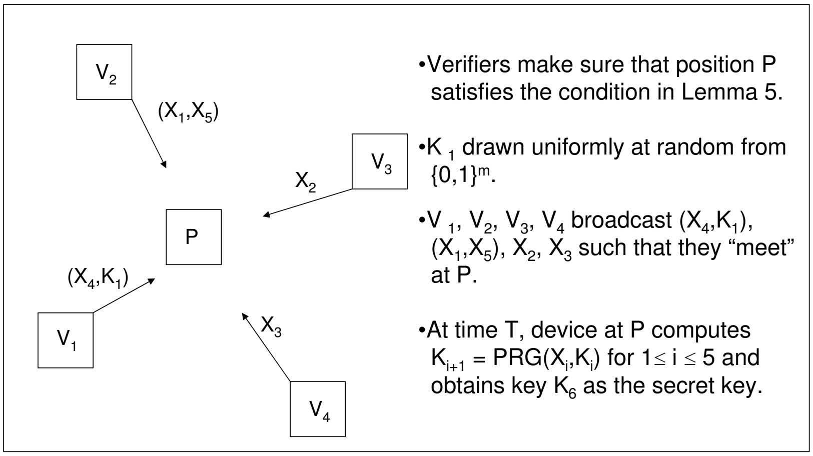

V2
(X1,X5)
V3
X2
P
(X4,K1)
V1
X3
V4
•Verifiers make sure that position P satisfies the condition in Lemma 5.
•K 1 drawn uniformly at random from {0,1}m.
•V 1, V2, V3, V4 broadcast (X4,K1), (X1,X5), X2, X3 such that they "meet" at P.
•At time T, device at P computes Ki+1 = PRG(Xi,Ki) for 1≤ i ≤ 5 and obtains key K6 as the secret key.

Fig. 2. Position based key exchange in 3-Dimensions

to both these types of adversaries together, the final key K_3 remains uniform. Indeed it turns out that this intuition is sound and the above is a secure one dimensional information theoretic position based key exchange protocol.

For three dimensions, we have the prover to be inside the tetrahedron defined by the four verifiers. Now, one can similarly try to extend the three-dimensional information theoretic secure positioning protocol presented earlier to achieve three-dimensional information theoretic position based key exchange. Simply add a fourth long string X_4 to be sent by V_1 in the natural way. However, it turns out that the above idea is not sufficient because of the fact that there might be adversaries (far) outside this tetrahedron trying to compute the key exchanged between the verifiers and an honest prover. In the case of secure positioning, such adversaries would be too late in sending their response to the verifiers (there is no honest prover to aid these adversaries). However, the key exchange scenario requires that once the verifiers and the honest prover get a shared key after running the protocol, this key should be uniform to the adversaries even at a much later point in time.

In contrast to what the intuition might suggest, the first problem we face is that there are certain regions in the tetrahedron defined by the verifiers such that if the claimed position P lies within one of these regions, there exists points, other than the point P, in the three dimensional space (but outside the tetrahedron) where the messages broadcast by the four verifiers all meet simultaneously. Thus, if there is an adversary located at such a point, it can compute the final key shared between the verifiers and the honest prover simply by following the algorithm of the honest prover. To overcome this problem, we characterize such regions of the tetrahedron (we further show that the remaining region is a still a large fraction of the tetrahedron) and exclude them from the area from which position claims are accepted (refer the full version for the Lemma characterizing such regions and for further details). That is, given an area from which position claims need to be accepted, our lemma depicts the acceptable positioning of the verifiers so that they can verify the position claims from that area.

The second main problem that arises is that even if the messages broadcast by the verifiers do not all meet at a single point (other than P), there of course could be multiple colluding adversaries which utilize different information available at multiple different points at different time instances to try to compute the final key. Indeed, it can be shown that there is in fact an explicit attack on the protocol discussed earlier (that is, the protocol resulting from a natural extension of our three-dimensional secure positioning protocol where the verifiers broadcast four long strings) which allows multiple colluding adversaries to completely recover the key exchanged between the verifiers and an honest prover. To solve the above problem, we introduce a fifth long string in a similar way as before. Introducing this fifth long string allows us to construct a geometric argument, along with a reduction argument relying on techniques from [14], that multiple colluding adversaries do not have sufficient information, and our security proofs go through. Our final protocol is given in Figure 2. Our security proofs are a careful combination of the following two components:

– A geometric argument which rules out a "nice" way for adversaries to recover the final key exchanged. In other words, very roughly, there does not exist a strategy for multiple colluding adversaries to perform the operation $K_{i+1} = \mathrm{PRG}_i(X_i, K_i)$ in sequence for each $i \in [5]$ to recover the final key K_6.
– A reduction argument relying on the techniques from [14] to prove the final security of our protocol. In more detail, given the above geometric argument, if there exists an adversarial strategy that can distinguish the final key K_6 from uniform in our protocol, then we can construct an adversarial strategy to contradict the security guarantees of an intrusion resilient secret sharing scheme (as defined and constructed in [14]).

All details of our protocol and the security proofs are given in the full version [7] of this paper. The completeness of the above protocol described relies on the assumption that the verifiers can store the long strings they generated to be able to compute the final key K_6 themselves. In the full version, we show that, as with the case of secure positioning, this assumption can be relaxed by using the same secret sharing technique introduced in Section 5.

Acknowledgments. We thank Yevgeniy Dodis for interesting discussions.

References

1. Aumann, Y., Rabin, M.O.: Information theoretically secure communication in the limited storage space model. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 65–79. Springer, Heidelberg (1999)
2. Brands, S., Chaum, D.: Distance-bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
3. Bussard, L.: Trust Establishment Protocols for Communicating Devices. Ph.D thesis, Eurecom-ENST (2004)

4. Cachin, C., Maurer, U.M.: Unconditional security against memory-bounded adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 292–306. Springer, Heidelberg (1997)
5. Capkun, S., Cagalj, M., Srivastava, M.: Secure localization with hidden and mobile base stations. In: IEEE INFOCOM (2006)
6. Capkun, S., Hubaux, J.-P.: Secure positioning of wireless devices with application to sensor networks. In: IEEE INFOCOM, pp. 1917–1928 (2005)
7. Chandran, N., Goyal, V., Moriarty, R., Ostrovsky, R.: Position based cryptography. Cryptology ePrint Archive (2009), `http://eprint.iacr.org/2009/`
8. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
9. Ding, Y.Z.: Oblivious transfer in the bounded storage model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 155–170. Springer, Heidelberg (2001)
10. Ding, Y.Z.: Error correction in the bounded storage model. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 578–599. Springer, Heidelberg (2005)
11. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
12. Dziembowski, S.: On forward-secure storage. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 251–270. Springer, Heidelberg (2006)
13. Dziembowski, S., Maurer, U.M.: On generating the initial key in the bounded-storage model. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 126–137. Springer, Heidelberg (2004)
14. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: FOCS 2007: Proceedings of the 48th Annual IEEE Foundations of Computer Science (2007)
15. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
16. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
17. Lu, C.-J.: Encryption against storage-bounded adversaries from on-line strong extractors. J. Cryptology 17(1), 27–42 (2004)
18. Maurer, U.M.: Conditionally-perfect secrecy and a provably-secure randomized cipher. J. Cryptology 5(1), 53–66 (1992)
19. Moran, T., Shaltiel, R., Ta-Shma, A.: Non-interactive timestamping in the bounded storage model. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 460–476. Springer, Heidelberg (2004)
20. Sastry, N., Shankar, U., Wagner, D.: Secure verification of location claims. In: WiSe 2003: Proceedings of the 2003 ACM workshop on Wireless security, pp. 1–10 (2003)
21. Singelee, D., Preneel, B.: Location verification using secure distance bounding protocols. In: IEEE Conference on Mobile Adhoc and Sensor Systems Conference (2005)
22. Vadhan, S.P.: Constructing locally computable extractors and cryptosystems in the bounded-storage model. J. Cryptology 17(1), 43–77 (2004)
23. Vora, A., Nesterenko, M.: Secure location verification using radio broadcast. In: Higashino, T. (ed.) OPODIS 2004. LNCS, vol. 3544, pp. 369–383. Springer, Heidelberg (2005)
24. Zhang, Y., Liu, W., Fang, Y., Wu, D.: Secure localization and authentication in ultra-wideband sensor networks. IEEE Journal on Selected Areas in Communications 24, 829–835 (2006)

Improving the Security of Quantum Protocols via Commit-and-Open

Ivan Damgård[1], Serge Fehr[2,⋆], Carolin Lunemann[1,⋆⋆], Louis Salvail[3,⋆⋆⋆], and Christian Schaffner[2,†]

[1] DAIMI, Aarhus University, Denmark
{ivan,carolin}@cs.au.dk
[2] Centrum Wiskunde & Informatica (CWI) Amsterdam, The Netherlands
{s.fehr,c.schaffner}@cwi.nl
[3] Université de Montréal (DIRO), QC, Canada
salvail@iro.umontreal.ca

Abstract. We consider two-party quantum protocols starting with a transmission of some random BB84 qubits followed by classical messages. We show a general "compiler" improving the security of such protocols: if the original protocol is secure against an "almost honest" adversary, then the compiled protocol is secure against an arbitrary computationally bounded (quantum) adversary. The compilation preserves the number of qubits sent and the number of rounds up to a constant factor. The compiler also preserves security in the bounded-quantum-storage model (BQSM), so if the original protocol was BQSM-secure, the compiled protocol can only be broken by an adversary who has large quantum memory *and* large computing power. This is in contrast to known BQSM-secure protocols, where security breaks down completely if the adversary has larger quantum memory than expected. We show how our technique can be applied to quantum identification and oblivious transfer protocols.

1 Introduction

We consider two-party quantum protocols for mutually distrusting players Alice and Bob. Such protocols typically start by Alice sending n random BB84 qubits to Bob who is supposed to measure them. Then some classical exchange of messages follows. Several protocols following this pattern have been proposed, implementing Oblivious Transfer (OT), Commitment, and Password-Based Identification [1,4,7,8].

⋆ Supported by the Dutch Organization for Scientific Research (NWO).
⋆⋆ Supported by the MOBISEQ project funded by NABIIT, Denmark.
⋆⋆⋆ Supported by the QUSEP project of the Danish Natural Science Research Council, and by the QuantumWorks Network.
† Supported by EU fifth framework project QAP IST 015848 and the NWO VICI project 2004-2009.

In more details, the first step of the protocol consists of Alice choosing random binary strings $x = x_1, ..., x_n$ and $\theta = \theta_1, ..., \theta_n$. She then prepares n particles where x_i is encoded in the state of the i'th particle using basis θ_i. Bob chooses a basis string $\hat{\theta} = \hat{\theta}_1, .., \hat{\theta}_n$ and measures the i'th particle in basis $\hat{\theta}_i$. If Bob plays honestly, he learns x_i whenever $\hat{\theta}_i = \theta_i$ and else gets a random independent result.

Protocols of the form we consider here are typically unconditionally secure against cheating by Alice, but can (in their basic form) be broken easily by Bob, if he does not measure the qubits immediately. This is because the protocol typically asks Alice to reveal θ at a later stage, and Bob can then measure the qubits with $\hat{\theta} = \theta$ and learn more information than he was supposed to.

In this paper, we show a general "compiler" that can be used to improve security against such an attack. We assume that the original protocol implements some two-party functionality $\mathcal{F}$ with statistical security against Bob if he is *benign*, meaning that he treats the qubits "almost honestly", a notion we make more precise below. Then we show that the compiled protocol also implements $\mathcal{F}$, but now with security against *any* computationally bounded (quantum) Bob (note that we cannot in general obtain unconditional security against both Alice and Bob, not even using quantum communication [13]). The compiled protocol preserves unconditional security against Alice and has the same number of transmitted qubits and rounds as the original one up to a constant factor.

By benign behavior of Bob, we mean that after having received the qubits, two conditions are satisfied: First, Bob's quantum storage is essentially of size zero (note that it would be exactly zero if he had measured the qubits). Second, there exists a basis string $\hat{\theta}$ such that the uncertainty about x is essentially as it would be if Bob had really measured in bases $\hat{\theta}$, namely 1 bit for every position where $\hat{\theta}$ differs from θ.

Thus, with our compiler, one can build a protocol for any two-party functionality by designing a protocol that only has to be secure if Bob is benign. We note that proofs for known protocols typically go through under this assumption. For instance, our compiler can easily be applied to the quantum identification protocols of [7] and the OT protocol of [1].

The compiler is based on a computational assumption; namely we assume the existence of a classical commitment scheme with some special properties, similar to the commitment schemes used in [5] but with an additional extraction property, secure against a quantum adversary. A good candidate is the cryptosystem of Regev [16]. For efficiency, we use a common reference string which allows us to use Regev's scheme in a simple way and, since it is relatively efficient, we get a protocol that is potentially practical. It is possible to generate the reference string from scratch, but this requires a more complicated non-constant round protocol [9].

The reader may ask whether it is really interesting to improve the security of quantum protocols for classical tasks such as identification or OT using a computational assumption. Perhaps it would be a more practical approach to use the same assumption to build *classical* protocols for the same tasks, secure against quantum attacks? To answer this, it is important to point out that our compiler also preserves security in the bounded-quantum-storage model (BQSM) [6], and this feature allows us to get security properties that classical protocols cannot achieve. In the BQSM, one assumes that Bob can only keep in his quantum memory a limited number of qubits received from Alice. With current state of the art, it is much easier to transmit and measure qubits than it is to store them for a non-negligible time, suggesting that the BQSM and the subsequently proposed noisy-quantum-storage model [20] are reasonable. On the other hand, if the assumption fails and the adversary can perfectly store all qubits sent, the known protocols can be easily broken. In contrast, by applying our compiler, one obtains new protocols where the adversary must have large quantum storage *and* large computing power to break the protocol.[1]

The basic technique we use to construct the compiler was already suggested in connection with the first quantum OT protocol from [1]: we try to force Bob to measure by asking him to commit (using a classical scheme) to all his basis choices and measurement results, and open some of them later. While classical intuition suggests that the commitments should force Bob to measure (almost) all the qubits, it has proved very tricky to show that the approach really works against a quantum adversary. In fact, it was previously very unclear what exactly the commit-and-open approach forces Bob to do. Although some partial results for OT have been shown [21,2], the original OT protocol from [1] has never been proved secure for a concrete unconditionally hiding commitment scheme – which is needed to maintain unconditional security against Alice. In this paper, we develop new quantum information-theoretic tools (that may be of independent interest) to characterize what commit-and-open achieves in general, namely it forces Bob to be benign. This property allows us to apply the compiler to any two-party functionality and in particular to show that the OT from [1] is indeed secure when using an appropriate commitment scheme.

2 Preliminaries

We assume the reader to be familiar with the basic notation and concepts of quantum information processing [14]. In this paper, the computational or $+$-basis is defined by the pair $\{|0\rangle, |1\rangle\}$ (also written as $\{|0\rangle_+, |1\rangle_+\}$). The pair $\{|0\rangle_\times, |1\rangle_\times\}$ denotes the diagonal or $\times$-basis, where $|0\rangle_\times = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle_\times = (|0\rangle - |1\rangle)/\sqrt{2}$. We write $|x\rangle_\theta = |x_1\rangle_{\theta_1} \otimes \cdots \otimes |x_n\rangle_{\theta_n}$ for the n-qubit state where string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ is encoded in bases $\theta = (\theta_1, \ldots, \theta_n) \in \{+, \times\}^n$. For $S \subseteq \{1, \ldots, n\}$ of size s, we denote by $\bar{S} := \{1, \ldots, n\} \backslash S$ the complement of S and define $x|_S \in \{0,1\}^s$ and $\theta|_S \in \{+, \times\}^s$ to be the restrictions $(x_i)_{i \in S}$ and $(\theta_i)_{i \in S}$, respectively. For two strings $x, y \in \{0,1\}^n$, we define the *Hamming distance* between x and y as $d_H(x,y) := |\{i : x_i \neq y_i\}|$.

[1] For the case of identification[7], the compiled protocol is not only secure against adversaries trying to impersonate Alice or Bob, but can also be made secure against man-in-the-middle attacks, where again the adversary must have large quantum storage and large computing power to break the protocol.

We use upper case letters for the random variables in the proofs that describe the respective values in the protocol. Given a bipartite quantum state ρ_{XE}, we say that X is *classical* if ρ_{XE} is of the form $\rho_{XE} = \sum_{x \in \mathcal{X}} P_X(x)|x\rangle\langle x| \otimes \rho_E^x$ for a probability distribution P_X over a finite set $\mathcal{X}$, i.e. the state of the quantum register E depends on the classical random variable X in the sense that E is in state ρ_E^x exactly if $X = x$. This naturally extends to states with two or more classical registers.

For a state ρ_{XE} as above, X is *independent* of register E if $\rho_{XE} = \rho_X \otimes \rho_E$, where $\rho_X = \sum_x P_X(x)|x\rangle\langle x|$ and $\rho_E = \sum_x P_X(x)\rho_E^x$. We also need to express that a random variable X is independent of a quantum state E *when given a random variable* Y. Independence means that when given Y, the state E gives no additional information on X. Formally, adopting the notion introduced in [7], we require that ρ_{XYE} equals $\rho_{X\leftrightarrow Y\leftrightarrow E}$, where the latter is defined as

$$\rho_{X\leftrightarrow Y\leftrightarrow E} := \sum_{x,y} P_{XY}(x,y)|x\rangle\langle x| \otimes |y\rangle\langle y| \otimes \rho_E^y\,,$$

where $\rho_E^y := \sum_x P_{X|Y}(x|y)\rho_E^{x,y}$. In other words, $\rho_{XYE} = \rho_{X\leftrightarrow Y\leftrightarrow E}$ precisely if $\rho_E^{x,y} = \rho_E^y$ for all x and y.

Full (conditional) independence is often too strong a requirement, and it usually suffices to be "close" to such a situation. Closeness of two states ρ and σ is measured in terms of their trace distance $\delta(\rho,\sigma) = \frac{1}{2}\operatorname{tr}(|\rho - \sigma|)$, where for any operator A, $|A|$ is defined as $|A| := \sqrt{AA^\dagger}$.

A quantum algorithm consists of a family $\{C_n\}_{n\in\mathbb{N}}$ of quantum circuits and is said to run in *polynomial time*, if the number of gates of C_n is polynomial in n. Two families of quantum states $\{\rho_n\}_{n\in\mathbb{N}}$ and $\{\sigma_n\}_{n\in\mathbb{N}}$ are called *quantum-computationally indistinguishable*, denoted $\rho \overset{q}{\approx} \sigma$, if any polynomial-time quantum algorithm has negligible advantage (in n) of distinguishing ρ_n from σ_n. Analogously, we call them *statistically indistinguishable*, $\rho \overset{s}{\approx} \sigma$, if their trace distance $\delta(\rho_n,\sigma_n)$ is negligible in n.

Definition 2.1 (Min-Entropy). *The* min-entropy *of a random variable X with probability distribution P_X is defined as $H_\infty(X) := -\log\big(\max_x P_X(x)\big)$.*

Definition 2.2 (Max-Entropy). *The* max-entropy *of a density matrix ρ is defined as $H_0(\rho) := \log\big(\operatorname{rank}(\rho)\big)$.*

We will make use of the following properties of a pure state that can be written as a "small superposition" of basis vectors; the proof is given in the full version [3].

Lemma 2.3. *Let $|\varphi_{AE}\rangle \in \mathcal{H}_A \otimes \mathcal{H}_E$ be of the form $|\varphi_{AE}\rangle = \sum_{i\in J}\alpha_i|i\rangle|\varphi_E^i\rangle$, where $\{|i\rangle\}_{i\in I}$ is a basis of $\mathcal{H}_A$ and $J \subseteq I$. Then, the following holds.*

1. *Let $\tilde\rho_{AE} = \sum_{i\in J}|\alpha_i|^2|i\rangle\langle i| \otimes |\varphi_E^i\rangle\langle\varphi_E^i|$, and let W and $\tilde W$ be the outcome of measuring A of $|\varphi_{AE}\rangle$ respectively of $\tilde\rho_{AE}$ in some basis $\{|w\rangle\}_{w\in\mathcal{W}}$. Then,*
$$H_\infty(W) \geq H_\infty(\tilde W) - \log|J|\,.$$
2. *The reduced density matrix $\rho_E = \operatorname{tr}_A(|\varphi_{AE}\rangle\langle\varphi_{AE}|)$ has max-entropy*
$$H_0(\rho_E) \leq \log|J|\,.$$

3 Definition of Security

In order to define security of our two-party protocols, we follow the framework put forward by Fehr and Schaffner in [10]. We are interested in quantum protocols that implement *classical functionalities* such as oblivious transfer. Such primitives are often used as building blocks in more complicated classical (multi-party) protocols which implement advanced tasks. Therefore, it is natural to restrict our focus on quantum protocols that run in a classical environment and have classical in- and outputs. A two-party quantum protocol $\Pi = (\mathsf{A}_m, \mathsf{B}_m)$ consists of an infinite family of interactive quantum circuits for players Alice and Bob indexed by the security parameter m (in our case, m will also be the number of qubits transmitted). To ease notation, we often leave the dependence on m implicit. A classical non-reactive two-party *ideal functionality* $\mathcal{F}$ is given by a conditional probability distribution $P_{\mathcal{F}(U,V)|UV}$, inducing a pair of random variables $(X, Y) = \mathcal{F}(U, V)$ for every joint distribution of U and V. The definition of correctness of a protocol is straightforward.

Definition 3.1 (Correctness). *A protocol $\Pi = (\mathsf{A}, \mathsf{B})$* correctly implements *an ideal classical functionality $\mathcal{F}$, if for every distribution of the input values U and V, the resulting common output satisfies*

$$(U, V, (X, Y)) \overset{s}{\approx} (U, V, \mathcal{F}(U, V)).$$

Let us denote by $out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}}$ the joint output[2] of the "ideal-life" protocol, where Alice and Bob forward their inputs to $\mathcal{F}$ and output whatever they obtain from $\mathcal{F}$. And we write $out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}'}$ for the joint output of the execution of this protocol with a dishonest Bob with strategy $\hat{\mathsf{B}}'$ (and similarly for a dishonest Alice). Note that Bob's possibilities in the ideal world are very limited: he can produce some classical input V for $\mathcal{F}$ from his input quantum state V', and then he can prepare and output a quantum state Y' which might depend on $\mathcal{F}$'s classical reply Y.

3.1 Information-Theoretic Security

We define information-theoretic security using the real/ideal-world paradigm, which requires that by attacking a protocol in the real world the dishonest party cannot achieve (significantly) more than when attacking the corresponding functionality in the ideal world. To be consistent with the framework used in [10], we restrict the joint input state, consisting of a classical input to the honest party and a possibly quantum input to the dishonest party, to a special form: in case of a dishonest Bob (and correspondingly for a dishonest Alice), we require that Bob's input consists of a classical part Z and a quantum part V', such that the joint state $\rho_{UZV'}$ satisfies $\rho_{UZV'} = \rho_{U\leftrightarrow Z\leftrightarrow V'}$, i.e., that V' is correlated with Alice's input only via the classical Z. We call a joint input state of that form (respectively of the form $\rho_{U'ZV} = \rho_{U'\leftrightarrow Z\leftrightarrow V}$ in case of dishonest Alice) a

[2] We use a slightly different notation here than in [10]. Our notation $out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}}$ does not mention the name of the input registers and corresponds to $(\mathcal{F}_{\hat{\mathsf{A}},\hat{\mathsf{B}}})\rho_{UV}$ in [10].

legitimate input state. As shown in [10], this restriction on the input state leads to a meaningful security definition with a composition theorem that guarantees sequential composition within *classical* outer protocols. Furthermore, the results of Section 4 also hold when quantifying over all (possibly non-legitimate) joint input states.

Definition 3.2 (Unconditional security against dishonest Alice). *A protocol $\Pi = (\mathsf{A}, \mathsf{B})$ implements an ideal classical functionality $\mathcal{F}$ unconditionally securely against dishonest Alice, if for any real-world adversary A' there exists an ideal-world adversary $\hat{\mathsf{A}}'$ such that for any legitimate input state, it holds that the outputs in the real and ideal world are statistically indistinguishable, i.e.*

$$out^{\Pi}_{\mathsf{A}',\mathsf{B}} \overset{s}{\approx} out^{\mathcal{F}}_{\hat{\mathsf{A}}',\hat{\mathsf{B}}} .$$

Definition 3.3 (Unconditional security against dishonest Bob). *A protocol $\Pi = (\mathsf{A}, \mathsf{B})$ implements an ideal classical functionality $\mathcal{F}$ unconditionally securely against dishonest Bob, if for any real-world adversary B' there exists an ideal-world adversary $\hat{\mathsf{B}}'$ such that for any legitimate input state, it holds that the outputs in the real and ideal world are statistically indistinguishable, i.e.*

$$out^{\Pi}_{\mathsf{A},\mathsf{B}'} \overset{s}{\approx} out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}'} .$$

It has been shown in Theorem 5.1 in [10] that protocols fulfilling the above definitions compose sequentially as follows. For a classical real-life protocol Σ which makes at most k oracle calls to functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$, it is guaranteed that whatever output Σ produces, the output produced when the oracle calls are replaced by ε-secure protocols is at distance at most $O(k\varepsilon)$.

3.2 Computational Security in the CRS Model

One can define security against a computationally bounded dishonest Bob analogously to information-theoretic security with the two differences that the input given to the parties has to be sampled by an efficient quantum algorithm and that the output states should be computationally indistinguishable.

In the common-reference-string (CRS) model, all participants in the real-life protocol $\Pi_{\mathsf{A},\mathsf{B}}$ have access to a classical public string ω which is chosen before any interaction starts according to a distribution only depending on the security parameter. On the other hand, the participants in the "ideal-life" protocol $\mathcal{F}_{\hat{\mathsf{A}},\hat{\mathsf{B}}}$ interacting only with the ideal functionality do not make use of the string ω. Hence, an ideal-world adversary $\hat{\mathsf{B}}'$, that operates by simulating the real world to the adversary B', is free to choose ω in any way he wishes.

In order to define computational security against a dishonest Bob in the CRS model, we consider a polynomial-size quantum circuit, called *input sampler*, which takes as input the security parameter m and the CRS ω (chosen according to its distribution) and produces the input state $\rho_{UZV'}$; U is Alice's classical input to the protocol, and Z and V' denote the respective classical and quantum information given to dishonest Bob. We call the input sampler *legitimate* if $\rho_{UZV'} = \rho_{U \leftrightarrow Z \leftrightarrow V'}$.

In the following and throughout the article, we let $\mathfrak{B}_{\text{poly}}$ be the family of all *polynomial-time* quantum strategies for dishonest Bob B'.

Definition 3.4 (Computational security against dishonest Bob). *A protocol $\Pi = (\mathsf{A}, \mathsf{B})$ implements an ideal classical functionality $\mathcal{F}$ computationally securely against dishonest Bob, if for any real-world adversary $\mathsf{B}' \in \mathfrak{B}_{\text{poly}}$ who has access to the common reference string ω, there exists an ideal-world adversary $\hat{\mathsf{B}}' \in \mathfrak{B}_{\text{poly}}$ not using ω such that for any efficient legitimate input sampler, it holds that the outputs in the real and ideal world are q-indistinguishable, i.e.*

$$out^{\Pi}_{\mathsf{A},\mathsf{B}'} \overset{q}{\approx} out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}'} .$$

In the full version [3], we show that also the computational security definition, as given here, allows for (sequential) composition of quantum protocols into classical outer protocols.

4 Improving the Security via Commit-and-Open

4.1 Security Against Benign Bob

In this paper, we consider quantum two-party protocols that follow a particular but very typical construction design. These protocols consist of two phases, called *preparation* and *post-processing* phase, and are as specified in Figure 1. We call a protocol that follows this construction design a *BB84-type* protocol.

PROTOCOL Π

Preparation: A chooses $x \in_R \{0,1\}^n$ and $\theta \in_R \{+,\times\}^n$ and sends $|x\rangle_\theta$ to B, and B chooses $\hat{\theta} \in_R \{+,\times\}^n$ and obtains $\hat{x} \in \{0,1\}^n$ by measuring $|x\rangle_\theta$ in basis $\hat{\theta}$.

Post-processing: Arbitrary classical communication and local computations.

Fig. 1. Generic BB84-type quantum protocol Π

The following definition captures information-theoretic security against a somewhat mildly dishonest Bob who we call a *benign* (dishonest) Bob. Such a dishonest Bob is benign in that, in the preparation phase, he does not deviate too much from what he is supposed to do; in the post-processing phase though, he may be arbitrarily dishonest.

To make this description formal, we fix an arbitrary choice of θ and an arbitrary value for the classical information, z, which B' may obtain as a result of the preparation phase (i.e. $z = (\hat{\theta}, \hat{x})$ in case B' is actually honest). Let X denote the random variable describing the bit-string x, where we understand the distribution P_X of X to be conditioned on the fixed values of θ and z. Furthermore, let ρ_E be the state of B''s quantum register E after the preparation phase. Note that, still with fixed θ and z, ρ_E is of the form $\rho_E = \sum_x P_X(x)\rho_E^x$, where ρ_E^x is the state of B''s quantum register in case X takes on the value x. In general, the ρ_E^x's may be mixed, but we can think of them as being reduced pure states: $\rho_E^x = \mathrm{tr}_R(|\psi_{ER}^x\rangle\langle\psi_{ER}^x|)$ for a suitable register R and pure states $|\psi_{ER}^x\rangle$; we then call the state $\rho_{ER} = \sum_x P_X(x)|\psi_{ER}^x\rangle\langle\psi_{ER}^x|$ a *pointwise purification* (with respect to X) of ρ_E.

Obviously, in case B' is honest, X_i is fully random whenever $\theta_i \neq \hat{\theta}_i$, so that $H_\infty\big(X|_I \,\big|\, X|_{\bar{I}} = x|_{\bar{I}}\big) = d_H\big(\theta|_I, \hat{\theta}|_I\big)$ for every $I \subseteq \{1,\ldots,n\}$ and every $x|_I$, and B' does not store any non-trivial quantum state so that R is "empty" and $H_0(\rho_{ER}) = H_0(\rho_E) = 0$. A benign Bob B' is now specified to behave close-to-honestly in the preparation phase: he produces an auxiliary output $\hat{\theta}$ after the preparation phase, and given this output, we are in a certain sense close to the ideal situation where Bob really measured in basis $\hat{\theta}$ as far as the values of $H_\infty\big(X|_I \,\big|\, X|_{\bar{I}} = x|_{\bar{I}}\big)$ and $H_0(\rho_{ER})$ are concerned.[3] We now make this precise:

Definition 4.1 (Unconditional security against *benign* Bob). *A BB84-type quantum protocol Π securely implements $\mathcal{F}$ against a β-*benign *Bob for some parameter $\beta \geq 0$, if it securely implements $\mathcal{F}$ according to Definition 3.3, with the following two modifications:*

1. *The quantification is over all B' with the following property: after the preparation phase B' either aborts, or else produces an auxiliary output $\hat{\theta} \in \{+,\times\}^n$. Moreover, the joint state of A,B' (after $\hat{\theta}$ has been output) is statistically indistinguishable from a state for which it holds that for any fixed values of θ, $\hat{\theta}$ and z, for any subset $I \subseteq \{1,\ldots,n\}$, and for any $x|_{\bar{I}}$*
$$H_\infty\big(X|_I \,\big|\, X|_{\bar{I}} = x|_{\bar{I}}\big) \geq d_H\big(\theta|_I, \hat{\theta}|_I\big) - \beta n \qquad \textit{and} \qquad H_0\big(\rho_{ER}\big) \leq \beta n \quad (1)$$
where ρ_{ER} is the pointwise purification of ρ_E with respect to X.
2. *$\hat{\mathsf{B}}'$'s running-time is polynomial in the running-time of B'.*

4.2 From Benign to Computational Security

We show a generic compiler which transforms any BB84-type protocol into a new quantum protocol for the same task. The compiler achieves that if the original protocol is unconditionally secure against dishonest Alice and unconditionally secure against *benign* Bob, then the compiled protocol is still unconditionally secure against dishonest Alice and it is *computationally secure* against *arbitrary* dishonest Bob.

The idea behind the construction of the compiler is to incorporate a commitment scheme and force Bob to behave benignly by means of a commit-and-open

[3] The reason why we consider the *pointwise purification* of ρ_E is to prevent Bob from artificially blowing up $H_0(\rho_{ER})$ by locally generating a large mixture or storing an unrelated mixed input state.

procedure. Figure 2 shows the compilation of an arbitrary BB84-type protocol Π. The quantum communication is increased from n to $m = n/(1-\alpha)$ qubits, where $0 < \alpha < 1$ is some additional parameter that can be arbitrarily chosen. The compiled protocol also requires 3 more rounds of interaction.

Protocol $\mathcal{C}^\alpha(\Pi)$

Preparation: A chooses $x \in_R \{0,1\}^m$ and $\theta \in_R \{+,\times\}^m$ and sends $|x\rangle_\theta$ to B. Then, B chooses $\hat\theta \in_R \{+,\times\}^m$ and obtains $\hat x \in \{0,1\}^m$ by measuring $|x\rangle_\theta$ in basis $\hat\theta$.

Verification:
1. B commits to $\hat\theta$ and $\hat x$ position-wise: $c_i := \texttt{Commit}\big((\hat\theta_i, \hat x_i), r_i\big)$ with randomness r_i for $i = 1, \ldots, m$. He sends the commitments to A.
2. A sends a random test subset $T \subset \{1, \ldots, m\}$ of size αm. B opens c_i for all $i \in T$, and A checks that the openings were correct and that $x_i = \hat x_i$ whenever $\theta_i = \hat\theta_i$. If all tests are passed, A accepts, otherwise, she rejects and aborts.
3. The tested positions are discarded by both parties: A and B restrict x and θ, respectively $\hat\theta$ and $\hat x$, to $i \in \bar T$.

Post-processing: As in Π (with $x, \theta, \hat x$ and $\hat\theta$ restricted to the positions $i \in \bar T$).

Fig. 2. Compiled protocol $\mathcal{C}^\alpha(\Pi)$

We need to specify what kind of commitment scheme to use. In order to preserve unconditional security against dishonest Alice, the commitment scheme needs to be unconditionally hiding, and so can at best be computationally binding. However, for a plain computationally binding commitment scheme, we do not know how to reduce the computational security of $\mathcal{C}^\alpha(\Pi)$ against dishonest Bob to the computational binding property of the commitment scheme.[4] Therefore, we use a commitment scheme with additional properties: we require a *keyed* commitment scheme $\texttt{Commit}_{\texttt{pk}}$, where the corresponding public key $\texttt{pk}$ is generated by one of two possible key-generation algorithms: $\mathcal{G}_{\texttt{H}}$ or $\mathcal{G}_{\texttt{B}}$. For a key $\texttt{pkH}$ generated by $\mathcal{G}_{\texttt{H}}$, the commitment scheme $\texttt{Commit}_{\texttt{pkH}}$ is unconditionally hiding, whereas the other generator, $\mathcal{G}_{\texttt{B}}$, actually produces a key *pair* $(\texttt{pkB}, \texttt{sk})$, so that the secret key $\texttt{sk}$ allows to efficiently extract m from $\texttt{Commit}_{\texttt{pkB}}(m, r)$, and as such $\texttt{Commit}_{\texttt{pkB}}$ is unconditionally binding. Furthermore, we require $\texttt{pkH}$ and $\texttt{pkB}$ to be computationally indistinguishable, even against quantum attacks. We call such a commitment scheme a *dual-mode* commitment scheme.[5] As a candidate for implementing such a system, we propose the public-key encryption scheme of Regev [16], which is based on a worst-case lattice assumption and is not known

[4] Classically, this would be done by a rewinding argument, but this fails to work for a quantum Bob.

[5] The notions of dual-mode *cryptosystems* and of meaningful/meaningless encryptions, as introduced in [15] and [12], are similar in spirit but differ slightly technically.

to be breakable even by (efficient) quantum algorithms. Regev does not explicitly state that the scheme has the property we need, but this is implicit in his proof that the underlying computational assumption implies semantic security. For simplicity and efficiency, we consider the common-reference-string model, and we assume the key pkB for the commitment scheme, generated according to $\mathcal{G}_{\mathtt{B}}$, to be contained in the CRS. We sketch in Section 6 how to avoid the CRS model, at the cost of a non constant-round construction where the parties generate the CRS jointly by means of a coin-tossing protocol (see [9] for details).

We sometimes write $\mathcal{C}^{\alpha}_{\mathtt{pkH}}(\Pi)$ for the compiled protocol $\mathcal{C}^{\alpha}(\Pi)$ to stress that a key pkH produced by $\mathcal{G}_{\mathtt{H}}$ is used for the dual-mode commitment scheme, and we write $\mathcal{C}^{\alpha}_{\mathtt{pkB}}(\Pi)$ when a key pkB produced by $\mathcal{G}_{\mathtt{B}}$ is used instead.

Theorem 4.2. *Let Π be a BB84-type protocol, unconditionally secure against dishonest Alice and against β-benign Bob for some constant $\beta > 0$. Consider the compiled protocol $\mathcal{C}^{\alpha}(\Pi)$ for an arbitrary $\alpha > 0$, where the commitment scheme is instantiated by a dual-mode commitment scheme as described above. Then, $\mathcal{C}^{\alpha}(\Pi)$ is unconditionally secure against dishonest Alice and computationally secure against dishonest Bob in the CRS model.*

We now prove this theorem, which assumes noise-free quantum communication; we explain in Section 4.4 how to generalize it for a noisy quantum channel. Correctness and unconditional security against dishonest Alice are obvious; the latter is due to the unconditional hiding property of the commitment scheme. As for computational security against dishonest Bob, according to Definition 3.4, we need to prove that for every real-world adversary $\mathsf{B}' \in \mathfrak{B}_{\text{poly}}$ attacking $\mathcal{C}^{\alpha}(\Pi)$, there exists a suitable ideal-world adversary $\hat{\mathsf{B}}' \in \mathfrak{B}_{\text{poly}}$ attacking $\mathcal{F}$ such that

$$out_{\mathsf{A},\mathsf{B}'}^{\mathcal{C}^{\alpha}(\Pi)} \overset{q}{\approx} out_{\hat{\mathsf{A}},\hat{\mathsf{B}}'}^{\mathcal{F}} .$$

First, note that by the computational indistinguishability of pkH and pkB,

$$out_{\mathsf{A},\mathsf{B}'}^{\mathcal{C}^{\alpha}(\Pi)} = out_{\mathsf{A},\mathsf{B}'}^{\mathcal{C}^{\alpha}_{\mathtt{pkH}}(\Pi)} \overset{q}{\approx} out_{\mathsf{A},\mathsf{B}'}^{\mathcal{C}^{\alpha}_{\mathtt{pkB}}(\Pi)} . \tag{2}$$

Then, we construct an adversary $\mathsf{B}'_{\circ} \in \mathfrak{B}_{\text{poly}}$ who attacks the unconditional security against benign Bob of protocol Π, and which satisfies

$$out_{\mathsf{A},\mathsf{B}'}^{\mathcal{C}^{\alpha}_{\mathtt{pkB}}(\Pi)} = out_{\mathsf{A}_{\circ},\mathsf{B}'_{\circ}}^{\Pi} , \tag{3}$$

where $\mathsf{A}_{\circ}$ honestly executes Π. We define $\mathsf{B}'_{\circ}$ in the following way. Consider the execution of $\mathcal{C}^{\alpha}(\Pi)$ between A and B'. We split A into two players $\mathsf{A}_{\circ}$ and $\tilde{\mathsf{A}}$, where we think of $\tilde{\mathsf{A}}$ as being placed in between $\mathsf{A}_{\circ}$ and B', see Figure 3. $\mathsf{A}_{\circ}$ plays honest Alice's part of Π while $\tilde{\mathsf{A}}$ acts as follows: It receives n qubits from $\mathsf{A}_{\circ}$, produces $\alpha n/(1-\alpha)$ random BB84 qubits of its own and interleaves them randomly with those received and sends the resulting $m = n/(1-\alpha)$ qubits to B'. It then does the verification step of $\mathcal{C}^{\alpha}(\Pi)$ with B', asking to have commitments corresponding to its own qubits opened. If this results in accept, it lets $\mathsf{A}_{\circ}$ finish the protocol with B'. Note that the pair $(\mathsf{A}_{\circ}, \tilde{\mathsf{A}})$ does exactly the same as A; however, we can also move the actions of $\tilde{\mathsf{A}}$ to Bob's side, and define $\mathsf{B}'_\circ$ as follows. $\mathsf{B}'_\circ$ samples $(\mathtt{pkB}, \mathtt{sk})$ according to $\tilde{\mathcal{G}}_\mathtt{B}$ and executes Π with A by locally running $\tilde{\mathsf{A}}$ and B', using $\mathtt{pkB}$ as CRS. If $\tilde{\mathsf{A}}$ accepts the verification then $\mathsf{B}'_\circ$ outputs $\hat{\theta} \in \{0,1\}^n$ (as required from a *benign* Bob), obtained by decrypting the unopened commitments with the help of $\mathtt{sk}$; else, $\mathsf{B}'_\circ$ aborts at this point. It is now clear that Equation (3) holds: exactly the same computation takes place in both "experiments", the only difference being that they are executed partly by different entities. The last step is to show that

$$out^{\Pi}_{\mathsf{A}_\circ,\mathsf{B}'_\circ} \overset{s}{\approx} out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}'}\,, \tag{4}$$

for some $\hat{\mathsf{B}}'$. It is clear that the theorem follows from (2) - (4) together.

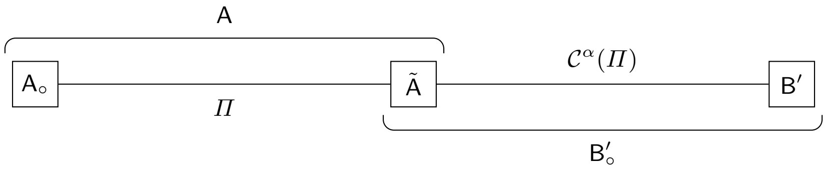


Fig. 3. Constructing an attacker $\mathsf{B}'_\circ$ against Π from an attacker B' against $\mathcal{C}^\alpha(\Pi)$

Now (4) actually claims that $\hat{\mathsf{A}}, \hat{\mathsf{B}}'$ successfully simulate $\mathsf{A}_\circ$ and $\mathsf{B}'_\circ$ executing Π, and this claim follows by assumption of benign security of Π if we show that $\mathsf{B}'_\circ$ is β-benign according to Definition 4.1 for any $\beta > 0$. We show this in the following subsection, i.e., the joint state of $\mathsf{A}_\circ, \mathsf{B}'_\circ$ after the preparation phase is statistically indistinguishable from a state ρ_{Ideal} which satisfies the bounds (1) from Definition 4.1.

4.3 Completing the Proof: Bounding Entropy and Memory Size

First recall that $\mathsf{A}_\circ$ executing Π with $\mathsf{B}'_\circ$ can equivalently be thought of as A executing $\mathcal{C}^\alpha_{\mathtt{pkB}}(\Pi)$ with B'. Furthermore, a joint state of A, B' is clearly also a joint state of $\mathsf{A}_\circ, \mathsf{B}'_\circ$.

To show the existence of ρ_{Ideal} as promised above, it therefore suffices to show such a state for A, B'. In other words, we need to show that the execution of $\mathcal{C}^\alpha_{\mathtt{pkB}}(\Pi)$ with honest Alice A and arbitrarily dishonest Bob B' will, after verification, be close to a state where (1) holds. To show this closeness, we consider an equivalent EPR-pair version, where Alice creates m EPR pairs $(|00\rangle + |11\rangle)/\sqrt{2}$, sends one qubit in each pair to Bob and keeps the others in register A. Alice measures her qubits only when needed: she measures the qubits within T in Step 2 of the verification phase, and the remaining qubits at the end of the verification phase. With respect to the information Alice and Bob obtain, this EPR version is *identical* to the original protocol $\mathcal{C}^\alpha_{\mathtt{pkB}}(\Pi)$: the only difference is the point in time when Alice obtains certain information. Furthermore, we can also do the following modification without affecting (1). Instead of measuring her qubits in T in *her* basis $\theta|_T$, she measures them in *Bob's* basis $\hat{\theta}|_T$; however, she still verifies only whether $x_i = \hat{x}_i$ for those $i \in T$ with $\theta_i = \hat{\theta}_i$. Because the positions $i \in T$ with $\theta_i \neq \hat{\theta}_i$ are not used in the protocol at all, this change has no effect. As the commitment scheme is unconditionally binding if key pkB is used, Bob's basis $\hat{\theta}$ is well defined by his commitments (although hard to compute), even if Bob is dishonest. The resulting scheme is given in Figure 4.

PROTOCOL EPR-$\mathcal{C}^{\alpha}_{\mathtt{pkB}}(\Pi)$

Preparation: A prepares m EPR pairs and sends the second qubit in each pair to Bob while keeping the others in register $A = A_1 \cdots A_m$. B chooses $\hat{\theta} \in_R \{+,\times\}^m$ and obtains $\hat{x} \in \{0,1\}^m$ by measuring the received qubits in basis $\hat{\theta}$.

Verification:

1. B commits to $\hat{\theta}$ and $\hat{x}$ position-wise: $c_i := \mathtt{Commit}\big((\hat{\theta}_i, \hat{x}_i), r_i\big)$ with randomness r_i for $i = 1, \ldots, m$. He sends the commitments to A.
2. A sends a random test subset $T \subset \{1, \ldots, m\}$ of size αm. B opens c_i for all $i \in T$. A chooses $\theta \in_R \{+,\times\}^m$, measures registers A_i with $i \in T$ in basis $\hat{\theta}_i$ to obtain x_i, and she checks that the openings were correct and that $x_i = \hat{x}_i$ whenever $\theta_i = \hat{\theta}_i$ for $i \in T$. If all tests are passed, A accepts, otherwise, she rejects and aborts the protocol.
3. A measures the remaining registers in basis $\theta|_{\bar{T}}$ to obtain $x|_{\bar{T}}$. The tested positions are discarded by both parties: A and B restrict x and θ, respectively $\hat{\theta}$ and $\hat{x}$, to the positions $i \in \bar{T}$.

Post-processing: As in Π (with $x, \theta, \hat{x}$ and $\hat{\theta}$ restricted to the positions $i \in \bar{T}$).

Fig. 4. EPR version of $\mathcal{C}^{\alpha}_{\mathtt{pkB}}(\Pi)$

We consider an execution of the scheme from Figure 4 with an honest Alice A and a dishonest Bob B$'$, and we fix $\hat{\theta}$ and $\hat{x}$, determined by Bob's commitments. Let $|\varphi_{AE}\rangle \in \mathcal{H}_A \otimes \mathcal{H}_E$ be the state of the joint system right before Step 2 of the verification phase. Since in the end, we are anyway interested in the pointwise purification of Bob's state, we may indeed assume this state to be pure; if it is not, then we purify it and carry the purifying register R along with E. Clearly, if B$'$ had honestly done his measurements then $|\varphi_{AE}\rangle = |\hat{x}\rangle_{\hat{\theta}} \otimes |\varphi_E\rangle$ for some $|\varphi_E\rangle \in \mathcal{H}_E$. In this case, the quantum memory E would be empty: $H_0(|\varphi_E\rangle\langle\varphi_E|) = 0$. Moreover, X, obtained by measuring $A|_{\bar{T}}$ in basis $\theta|_{\bar{T}}$, would contain $d_H(\theta|_{\bar{T}}, \hat{\theta}|_{\bar{T}})$ random bits. We show that the verification phase enforces these properties, at least approximately in the sense of (1), for an arbitrary dishonest Bob B$'$.

In the following, $r_H(\cdot,\cdot)$ denotes the relative Hamming distance between two strings, i.e., the Hamming distance divided by their length. Recall that $T \subset \{1, \ldots, m\}$ is random subject to $|T| = \alpha m$. Furthermore, for a fixed $\hat{\theta}$ but a randomly chosen θ, the subset $T' = \{i \in T : \theta_i = \hat{\theta}_i\}$ is a random subset (of arbitrary size) of T. Let the random variable *Test* describe the choice of *test* $= (T, T')$ as specified above, and consider the state

$$\rho_{\mathit{Test}AE} = \rho_{\mathit{Test}} \otimes |\varphi_{AE}\rangle\langle\varphi_{AE}| = \sum_{\mathit{test}} P_{\mathit{Test}}(\mathit{test}) |\mathit{test}\rangle\langle\mathit{test}| \otimes |\varphi_{AE}\rangle\langle\varphi_{AE}|$$

consisting of the classical *Test* and the quantum state $|\varphi_{AE}\rangle$.

Lemma 4.3. *For any $\varepsilon > 0$, $\hat{x} \in \{0,1\}^m$ and $\hat{\theta} \in \{+,\times\}^m$, the state $\rho_{\mathit{Test}AE}$ is negligibly close (in m) to a state*

$$\tilde{\rho}_{\mathit{Test}AE} = \sum_{\mathit{test}} P_{\mathit{Test}}(\mathit{test}) |\mathit{test}\rangle\langle\mathit{test}| \otimes |\tilde{\varphi}^{\mathit{test}}_{AE}\rangle\langle\tilde{\varphi}^{\mathit{test}}_{AE}|$$

where for any test $= (T, T')$:

$$|\tilde{\varphi}^{\mathit{test}}_{AE}\rangle = \sum_{x \in B_{\mathit{test}}} \alpha^{\mathit{test}}_x |x\rangle_{\hat{\theta}} |\psi^x_E\rangle$$

for $B_{\mathit{test}} = \{x \in \{0,1\}^m \,|\, r_H(x|_{\bar{T}}, \hat{x}|_{\bar{T}}) \leq r_H(x|_{T'}, \hat{x}|_{T'}) + \varepsilon\}$ and arbitrary coefficients $\alpha^{\mathit{test}}_x \in \mathbb{C}$.

In other words, we are close to a situation where for *any* choice of T and T' and for *any* outcome $x|_T$ when measuring $A|_T$ in basis $\hat{\theta}|_T$, the relative error $r_H(x|_{T'}, \hat{x}|_{T'})$ gives an upper bound (which holds with probability 1) on the relative error $r_H(x|_{\bar{T}}, \hat{x}|_{\bar{T}})$ one would obtain by measuring the remaining subsystems A_i with $i \in \bar{T}$ in basis $\hat{\theta}_i$.

Proof. For any *test* we let $|\tilde{\varphi}^{\mathit{test}}_{AE}\rangle$ be the renormalized projection of $|\varphi_{AE}\rangle$ into the subspace $\mathrm{span}\{|x\rangle_{\hat{\theta}} \,|\, x \in B_{\mathit{test}}\} \otimes \mathcal{H}_E$ and let $|\tilde{\varphi}^{\mathit{test}\perp}_{AE}\rangle$ be the renormalized projection of $|\varphi_{AE}\rangle$ into the orthogonal complement, such that $|\varphi_{AE}\rangle = \varepsilon_{\mathit{test}} |\tilde{\varphi}^{\mathit{test}}_{AE}\rangle + \varepsilon^{\perp}_{\mathit{test}} |\tilde{\varphi}^{\mathit{test}\perp}_{AE}\rangle$ with $\varepsilon_{\mathit{test}} = \langle\tilde{\varphi}^{\mathit{test}}_{AE}|\varphi_{AE}\rangle$ and $\varepsilon^{\perp}_{\mathit{test}} = \langle\tilde{\varphi}^{\mathit{test}\perp}_{AE}|\varphi_{AE}\rangle$. By construction, $|\tilde{\varphi}^{\mathit{test}}_{AE}\rangle$ is of the form required in the statement of the lemma. A basic property of the trace norm of pure states gives

$$\delta\big(|\varphi_{AE}\rangle\langle\varphi_{AE}|, |\tilde{\varphi}^{\mathit{test}}_{AE}\rangle\langle\tilde{\varphi}^{\mathit{test}}_{AE}|\big) = \sqrt{1 - |\langle\tilde{\varphi}^{\mathit{test}}_{AE}|\varphi_{AE}\rangle|^2} = |\varepsilon^{\perp}_{\mathit{test}}| \,.$$

This last term corresponds to the square root of the probability, when given *test*, to observe a string $x \notin B_{\mathit{test}}$ when measuring subsystem A of $|\varphi_{AE}\rangle$ in basis $\hat{\theta}$. Furthermore, using elementary properties of the trace norm and Jensen's inequality gives

$$\begin{aligned}\delta\big(\rho_{\mathit{Test}AE}, \tilde{\rho}_{\mathit{Test}AE}\big)^2 &= \left(\sum_{\mathit{test}} P_{\mathit{Test}}(\mathit{test})\, \delta\big(|\varphi_{AE}\rangle\langle\varphi_{AE}|, |\tilde{\varphi}^{\mathit{test}}_{AE}\rangle\langle\tilde{\varphi}^{\mathit{test}}_{AE}|\big)\right)^2 \\ &= \left(\sum_{\mathit{test}} P_{\mathit{Test}}(\mathit{test})\, |\varepsilon^{\perp}_{\mathit{test}}|\right)^2 \leq \sum_{\mathit{test}} P_{\mathit{Test}}(\mathit{test})\, |\varepsilon^{\perp}_{\mathit{test}}|^2 \,,\end{aligned}$$

where the last term is the probability to observe a string $x \notin B_{test}$ when choosing *test* according to P_{Test} and measuring subsystem A of $|\varphi_{AE}\rangle$ in basis $\hat{\theta}$. This situation, though, is a classical sampling problem, for which it is well known that for any measurement outcome x, the probability (over the choice of *test*) that $x \notin B_{test}$ is negligible in m (see e.g. [11]). □

In combination with Lemma 2.3 on "small superpositions of product states", and writing h for the binary entropy function $h(\mu) = -\big(\mu \log(\mu) + (1-\mu)\log(1-\mu)\big)$ as well as using that $\big|\{y \in \{0,1\}^n \,|\, d_H(y,\hat{y}) \le \mu n\}\big| \le 2^{h(\mu)n}$ for any $\hat{y} \in \{0,1\}^n$ and $0 \le \mu \le \frac{1}{2}$, we can conclude the following (the proof is given in [3]).

Corollary 4.4. *Let $\tilde{\rho}_{TestAE}$ be of the form as in Lemma 4.3 (for given ε, $\hat{x}$ and $\hat{\theta}$). For any fixed test $= (T,T')$ and for any fixed $x|_T \in \{0,1\}^{\alpha m}$ with $err := r_H(x|_{T'}, \hat{x}|_{T'}) \le \frac{1}{2}$, let $|\psi_{AE}\rangle$ be the state to which $|\tilde{\varphi}_{AE}^{test}\rangle$ collapses when for every $i \in T$ subsystem A_i is measured in basis $\hat{\theta}_i$ and x_i is observed, where we understand A in $|\psi_{AE}\rangle$ to be restricted to the registers A_i with $i \in \bar{T}$. Finally, let $\sigma_E = \mathrm{tr}_A(|\psi_{AE}\rangle\langle\psi_{AE}|)$ and let the random variable X describe the outcome when measuring the remaining $n = (1-\alpha)m$ subsystems of A in basis $\theta|_{\bar{T}} \in \{+,\times\}^n$. Then, for any subset $I \subseteq \{1,\ldots,n\}$ and any $x|_I$,*[6]

$$H_\infty\big(X|_I \,\big|\, X|_{\bar{I}} = x|_{\bar{I}}\big) \ge d_H\big(\theta|_I, \hat{\theta}|_I\big) - h(err+\varepsilon)n \quad \textit{and} \quad H_0\big(\sigma_E\big) \le h(err+\varepsilon)n\,.$$

Thus, the number of errors between the measured $x|_{T'}$ and the given $\hat{x}|_{T'}$ gives us a bound on the min-entropy of the outcome when measuring the remaining subsystems of A, and on the max-entropy of the state of subsystem E.

The claim to be shown now follows by combining Lemma 4.3 and Corollary 4.4. Indeed, the ideal state ρ_{Ideal} we promised is produced by putting A and B' in the state $\tilde{\rho}_{TestAE}$ defined in Lemma 4.3, and running Steps 2 and 3 of the verification phase. This state is negligibly close to the real state since by Lemma 4.3 we were negligibly close to the real state before these operations. Corollary 4.4 guarantees that (1) is satisfied.

4.4 In the Presence of Noise

In the description of the compiler $\mathcal{C}^\alpha$ and in its analysis, we assumed the quantum communication to be noise-free. Indeed, if the quantum communication is noisy honest Alice is likely to reject an execution with honest Bob. It is straightforward to generalize the result to noisy quantum communication: In Step 2 in the verification phase of $\mathcal{C}^\alpha(\Pi)$, Alice rejects and aborts if the relative number of errors between x_i and $\hat{x}_i$ for $i \in T$ with $\theta_i = \hat{\theta}_i$ exceeds the error probability ϕ induced by the noise in the quantum communication by some small $\varepsilon' > 0$. By Hoeffding's inequality [11], this guarantees that honest Alice does not reject honest Bob except with exponentially small probability. Furthermore, proving the security of this "noise-resistant" compiler goes along the exact same lines

[6] Below, $\theta|_I$ (and similarly $\hat{\theta}|_I$) should be understood as first restricting the m-bit vector θ to $\bar{T}$, and then restricting the resulting n-bit vector $\theta|_{\bar{T}}$ to I: $\theta|_I := (\theta|_{\bar{T}})|_I$.

as for the original compiler. The only difference is that when applying Corollary 4.4, the parameter err has to be chosen as $err = \phi + \varepsilon'$, so that (1) holds for $\beta = h(err + \varepsilon) = h(\phi + \varepsilon' + \varepsilon)$ and thus the claim of Theorem 4.2 hold for any $\beta > h(\phi)$ (by choosing $\varepsilon, \varepsilon' > 0$ small enough). This allows us to generalize the results from the Section 5 to the setting of noisy quantum communication.

4.5 Bounded-Quantum-Storage Security

In this section we show that our compiler preserves security in the bounded-quantum-storage model (BQSM). In this model, one of the players (Bob in our case) is assumed be able to store only a limited number of qubits beyond a certain point in the protocol. BQSM-secure OT and identification protocols are known [4,7], but they can be efficiently broken if the memory bound does not hold. Therefore, by the theorem below, applying the compiler produces protocols with better security, namely the adversary needs large quantum storage *and* large computing power to succeed.

Consider a BB84-type protocol Π, and for a constant $0 < \gamma < 1$, let $\mathfrak{B}^{\gamma}_{\text{BQSM}}(\Pi)$ be the set of dishonest players B' that store only γn qubits after a certain point in Π, where n is the number of qubits sent initially. Protocol Π is said to be unconditionally secure against γ-BQSM Bob, if it satisfies Definition 3.3 with the restriction that the quantification is over all dishonest $\mathsf{B}' \in \mathfrak{B}^{\gamma}_{\text{BQSM}}(\Pi)$.

Theorem 4.5. *If Π is unconditionally secure against γ-BQSM Bob, then $\mathcal{C}^{\alpha}(\Pi)$ (for an $0 < \alpha < 1$) is unconditionally secure against $\gamma(1-\alpha)$-BQSM Bob.*

Proof. Exactly as in the proof of Theorem 4.2, given dishonest Bob B' attacking $\mathcal{C}^{\alpha}(\Pi)$, we construct dishonest Bob $\mathsf{B}'_\circ$ attacking the original protocol Π. The only difference here is that we let $\mathsf{B}'_\circ$ generate the CRS "correctly" as $\mathtt{pkH}$ sampled according to $\mathcal{G}_{\mathtt{H}}$. It follows by construction of $\mathsf{B}'_\circ$ that $out^{\mathcal{C}^{\alpha}(\Pi)}_{\mathsf{A},\mathsf{B}'} = out^{\Pi}_{\mathsf{A}_\circ,\mathsf{B}'_\circ}$. Also, it follows by construction of $\mathsf{B}'_\circ$ that if $\mathsf{B}' \in \mathfrak{B}^{\gamma(1-\alpha)}_{\text{BQSM}}(\mathcal{C}^{\alpha}(\Pi))$ then $\mathsf{B}'_\circ \in \mathfrak{B}^{\gamma}_{\text{BQSM}}(\Pi)$, since $\mathsf{B}'_\circ$ requires the same amount of quantum storage as B' but communicates an α-fraction fewer qubits. It thus follows that there exists $\hat{\mathsf{B}}'$ such that $out^{\Pi}_{\mathsf{A}_\circ,\mathsf{B}'_\circ} \overset{s}{\approx} out^{\mathcal{F}}_{\hat{\mathsf{A}},\hat{\mathsf{B}}'}$. This proves the claim. □

5 Applications

5.1 Oblivious Transfer

We discuss a protocol that securely implements one-out-of-two oblivious transfer of strings of length ℓ (i.e. 1-2 OT^{ℓ}). In 1-2 OT^{ℓ}, the sender A sends two l-bit strings s_0 and s_1 to the receiver B. B can choose which string to receive (s_k) but does not learn anything about the other one (s_{1-k}). On the other hand, A does not learn B's choice bit k. The protocol is almost identical to the 1-2 OT^{1} introduced in [1], but uses hash functions instead of parity values to mask the inputs s_0 and s_1. The resulting scheme, called $\mathtt{1\text{-}2\,QOT}^{\ell}$, is presented in Figure 5, where $\mathcal{F}$ denotes a suitable family of universal hash functions with range $\{0,1\}^{\ell}$ (as specified in [4]). We assume that $\ell = \lfloor \lambda n \rfloor$ for some constant $\lambda > 0$.

Protocol $\texttt{1-2 QOT}^\ell$:

Preparation: A chooses $x \in_R \{0,1\}^n$ and $\theta \in_R \{+,\times\}^n$ and sends $|x\rangle_\theta$ to B, and B chooses $\hat{\theta} \in_R \{0,1\}^n$ and obtains $\hat{x} \in \{0,1\}^n$ by measuring $|x\rangle_\theta$ in basis $\hat{\theta}$.

Post-processing:

1. A sends θ to B.
2. B partitions all positions $1 \le i \le n$ in two subsets according to his choice bit $k \in \{0,1\}$: the "good" subset $I_k := \{i : \theta_i = \hat{\theta}_i\}$ and the "bad" subset $I_{1-k} := \{i : \theta_i \neq \hat{\theta}_i\}$. B sends (I_0, I_1) to A.
3. A sends descriptions of $f_0, f_1 \in_R \mathcal{F}$ together with $m_0 := s_0 \oplus f_0(x|_{I_0})$ and $m_1 := s_1 \oplus f_1(x|_{I_1})$.
4. B computes $s_k = m_k \oplus f_k(\hat{x}|_{I_k})$.

Fig. 5. Protocol for String OT

Theorem 5.1. *Protocol* $\texttt{1-2 QOT}^\ell$ *is unconditionally secure against β-benign Bob for any $\beta < \frac{1}{8} - \frac{\lambda}{2}$.*

The proof and precise definition of OT security is given in [3]. By combining Theorem 5.1 with Theorem 4.2, and the results of [4] (realizing that the same analysis also applies to $\texttt{1-2 QOT}^\ell$) with Theorem 4.5, we obtain the following hybrid-security result.

Corollary 5.2. *Let $0 < \alpha < 1$ and $\lambda < \frac{1}{8}$. Then protocol $\mathcal{C}^\alpha(\texttt{1-2 QOT}^\ell)$ is computationally secure against dishonest Bob and unconditionally secure against $\gamma(1-\alpha)$-BQSM Bob with $\gamma < \frac{1}{4} - 2\lambda$.*

5.2 Password-Based Identification

We want to apply our compiler to the quantum password-based identification scheme from [7]. Such an identification scheme allows a user A to identify herself to server B by means of a common (possibly non-uniform and low-entropy) password $w \in \mathcal{W}$, such that dishonest A' cannot delude honest server B with probability better then trying to guess the password, and dishonest B' learns no information on A's password beyond trying to guessing it and learn whether the guess is correct or not.

In [7], using quantum-information-theoretic security definitions, the proposed identification scheme was proven to be unconditionally secure against arbitrary dishonest Alice and against quantum-memory-bounded dishonest Bob. In [10] it was then shown that these security definitions imply simulation-based security as considered here, with respect to the functionality $\mathcal{F}_{ID}$ given in Figure 6.[7]

[7] Actually, the definition and proof from [7] guarantees security only for a slightly weaker functionality, which gives some unfair advantage to dishonest A' in case she guesses the password correctly; however, as discussed in [10], the protocol from [7] does implement functionality $\mathcal{F}_{ID}$.

Functionality $\mathcal{F}_{ID}$: Upon receiving $w_A, w_B \in \mathcal{W}$ from user Alice and from server Bob, respectively, $\mathcal{F}_{ID}$ outputs the bit $y := (w_A \overset{?}{=} w_B)$ to Bob. In case Alice is dishonest, she may choose $w_A = \perp$ (where $\perp \notin \mathcal{W}$). For any choice of w_A the bit y is also output to dishonest Alice.

Fig. 6. The Ideal Password-Based Identification Functionality

We cannot directly apply our compiler to the identification scheme as given in [7], since it is *not* a BB84-type protocol. The protocol does start with a preparation phase in which Alice sends BB84 qubits to Bob, but Bob does not measure them in a random basis but in a basis determined by his password $w_B \in \mathcal{W}$; specifically, Bob uses as basis the encoding $\mathfrak{c}(w_B)$ of w_B with respect to a code $\mathfrak{c} : \mathcal{W} \to \{+, \times\}^n$ with "large" minimal distance. However, it is easy to transform the original protocol from [7] into a BB84-type protocol without affecting security: We simply let Bob apply a *random shift* κ to the code, which Bob only announces to Alice in the post-processing phase, and then Alice and Bob complete the protocol with the shifted code. The resulting protocol QID is described in Figure 7, where $\mathcal{F}$ and $\mathcal{G}$ are suitable families of (strongly) universal hash functions (we refer to [7] for the exact specifications). It is not hard to see that this modification does not affect security as proven in [7] (and [10]).

PROTOCOL QID :

Preparation: A chooses $x \in_R \{0,1\}^n$ and $\theta \in_R \{+, \times\}^n$ and sends $|x\rangle_\theta$ to B, and B chooses $\hat{\theta} \in_R \{0,1\}^n$ and obtains $\hat{x} \in \{0,1\}^n$ by measuring $|x\rangle_\theta$ in basis $\hat{\theta}$.

Post-processing:
1. B computes a string $\kappa \in \{+, \times\}^n$ such that $\hat{\theta} = \mathfrak{c}(w) \oplus \kappa$ (we think of $+$ as 0 and $\times$ as 1 so that $\oplus$ makes sense). He sends κ to A and we define $\mathfrak{c}'(w) := \mathfrak{c}(w) \oplus \kappa$.
2. A sends θ and $f \in_R \mathcal{F}$ to B. Both compute $I_w := \{i : \theta_i = \mathfrak{c}'(w)_i\}$.
3. B sends $g \in_R \mathcal{G}$.
4. A sends $z := f(x|_{I_w}) \oplus g(w)$ to B.
5. B accepts if and only if $z = f(\hat{x}|_{I_w}) \oplus g(w)$.

Fig. 7. Protocol for Secure Password-Based Identification

Theorem 5.3. *If the code $\mathfrak{c} : \mathcal{W} \to \{+, \times\}^n$ can correct at least δn errors in polynomial-time for a constant δ, then protocol* QID *is unconditionally secure against β-benign Bob for any $\beta < \frac{\delta}{4}$.*

Proof. For any given benign Bob B', we construct $\hat{\mathsf{B}}'$ as follows. $\hat{\mathsf{B}}'$ runs locally a copy of B' and simulates Alice's actions by running A faithfully except for the following modifications. After the preparation phase, $\hat{\mathsf{B}}'$ gets $\hat{\theta}$ and κ from B' and attempts to decode $\hat{\theta} \oplus \kappa$. If this succeeds, it computes w' such that $\mathfrak{c}(w')$ is the decoded codeword. Otherwise an arbitrary w' is chosen. Then, $\hat{\mathsf{B}}'$ submits w' as Bob's input w_B to $\mathcal{F}_{ID}$ and receives output $y \in \{0,1\}$. If $y = 1$ then $\hat{\mathsf{B}}'$ faithfully completes A's simulation using w' as w; else, $\hat{\mathsf{B}}'$ completes the simulation by using a random z' instead of z. In the end, $\hat{\mathsf{B}}'$ outputs whatever B' outputs.

We need to show that the state output by $\hat{\mathsf{B}}'$ (respectively B') above is statistically close to the state output by B' when executing $\mathtt{QID}$ with real A. Note that if $w' = w_A$, then the simulation of A is perfect and thus the two states are equal. If $w' \neq w_A$ then the simulation is not perfect: the real A would use $z = f(x|_{I_{w_A}}) \oplus g(w_A)$ instead of random z'. It thus suffices to argue that $f(x|_{I_w})$ is statistically close to random and independent of the view of B' for any fixed $w \neq w'$. Note that this is also what had to be proven in [7], but under a different assumption, namely that B' has bounded quantum memory, rather than that he is benign; nevertheless, we can recycle part of the proof.

Recall from the definition of a benign Bob that the common state after the preparation phase is statistically close to a state for which it is guaranteed that $H_\infty(X|_I) \geq d_H(\theta|_I, \hat{\theta}|_I) - \beta n$ for any $I \subseteq \{1,\dots,n\}$, and $H_0(\rho_{ER}) \leq \beta n$. By the closeness of these two states, switching from the real state to the "ideal" state (which satisfies these bounds) has only a negligible effect on the state output by $\hat{\mathsf{B}}'$; thus, we may assume these bounds to hold.

Now, if decoding of $\hat{\theta} \oplus \kappa$ succeeded, it is at Hamming distance at most δn from $\mathfrak{c}(w')$. Since the distance from here to the (distinct) codeword $\mathfrak{c}(w)$ is greater than $2\delta n$, we see that $\hat{\theta} \oplus \kappa$ is at least δn away from $\mathfrak{c}(w)$. The same is true if decoding failed, since then $\hat{\theta} \oplus \kappa$ is at least δn away from any codeword. It follows that $\mathfrak{c}'(w) = \mathfrak{c}(w) \oplus \kappa$ has Hamming distance at least δn from $\hat{\theta}$. Furthermore, for arbitrary $\varepsilon > 0$ and except with negligible probability, the Hamming distance between $\theta|_{I_w} = \mathfrak{c}'(w)|_{I_w}$ and $\hat{\theta}|_{I_w}$ is at least essentially $(\delta/2 - \varepsilon)n$. Therefore, we can conclude that $H_\infty(X|_{I_w}) \geq (\delta/2 - \varepsilon - \beta)n$ and $H_0(\rho_{ER}) \leq \beta n$. But now, if such bounds hold such that $H_\infty(X|_{I_w}) - H_0(\rho_{ER})$ is positive and linear in n, which is the case here by the choice of parameters, then we can step into the proof from [7] and conclude by privacy amplification [17] that z is close to random and independent of E. This finishes the proof. □

By combining Theorem 5.3 with Theorem 4.2, and the results of [7] with Theorem 4.5, we obtain the following hybrid-security result.

Corollary 5.4. *Let* $0 < \alpha < 1$ *and* $|\mathcal{W}| \leq 2^{\nu n}$. *If the code* $\mathfrak{c} : \mathcal{W} \to \{+,\times\}^n$ *can correct* δn *errors for a constant* $\delta > 0$ *in polynomial-time, then protocol* $\mathcal{C}^\alpha(\mathtt{QID})$ *is computationally secure against dishonest Bob and unconditionally secure against* $\gamma(1-\alpha)$*-BQSM Bob with* $\gamma < \frac{\delta}{2} - \nu$.

Families of codes as required in these results, correcting a constant fraction of errors efficiently and with constant information rate are indeed known, see [18].

In the full version [3], we discuss how to obtain hybrid security against *man-in-the-middle* attacks by means of incorporating the techniques used in [7] to obtain security in the BQSM against such attacks.

6 Doing without a Common Reference String

We can get rid of the CRS assumption by instead generating a reference string from scratch using a coin-flip protocol. In [9], such a coin-flip protocol is described and proved secure against quantum adversaries using Watrous' quantum rewinding method [19]. Note that for our compiler, we want the CRS to be an unconditionally hiding public key, and when using Regev's cryptosystem, a uniformly random string (as output by the coin-flip) does indeed determine such a key, except with negligible probability.

References

1. Bennett, C.H., Brassard, G., Crépeau, C., Skubiszewska, M.-H.: Practical quantum oblivious transfer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 351–366. Springer, Heidelberg (1991)
2. Crépeau, C., Dumais, P., Mayers, D., Salvail, L.: Computational collapse of quantum state with application to oblivious transfer. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 374–393. Springer, Heidelberg (2004)
3. Damgård, I.B., Fehr, S., Lunemann, C., Salvail, L., Schaffner, C.: Improving the security of quantum protocols (2009), `http://arxiv.org/abs/0902.3918`
4. Damgård, I.B., Fehr, S., Renner, R., Salvail, L., Schaffner, C.: A tight high-order entropic quantum uncertainty relation with applications. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 360–378. Springer, Heidelberg (2007)
5. Damgård, I.B., Fehr, S., Salvail, L.: Zero-knowledge proofs and string commitments withstanding quantum attacks. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 254–272. Springer, Heidelberg (2004)
6. Damgård, I.B., Fehr, S., Salvail, L., Schaffner, C.: Cryptography in the bounded quantum-storage model. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 449–458 (2005), `http://arxiv.org/abs/quant-ph/0508222v2`
7. Damgård, I.B., Fehr, S., Salvail, L., Schaffner, C.: Secure identification and QKD in the bounded-quantum-storage model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 342–359. Springer, Heidelberg (2007)
8. Damgård, I.B., Fehr, S., Salvail, L., Schaffner, C.: Cryptography in the bounded-quantum-storage model. SIAM Journal on Computing 37(6), 1865–1890 (2008)
9. Damgård, I.B., Lunemann, C.: Quantum-secure coin-flipping and applications (2009), `http://arxiv.org/abs/0903.3118`
10. Fehr, S., Schaffner, C.: Composing quantum protocols in a classical environment. In: Theory of Cryptography Conference (TCC). LNCS, vol. 5444, pp. 350–367. Springer, Heidelberg (2009)
11. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
12. Kol, G., Naor, M.: Games for exchanging information. In: TCC 2008. LNCS, vol. 4948, pp. 423–432. Springer, Heidelberg (2008)
13. Lo, H.-K.: Insecurity of quantum secure computations. Physical Review A 56(2), 1154–1162 (1997)
14. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)

15. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
16. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 84–93 (2005)
17. Renner, R., König, R.: Universally composable privacy amplification against quantum adversaries. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 407–425. Springer, Heidelberg (2005)
18. Sipser, M., Spielman, D.A.: Expander codes. IEEE Transactions on Information Theory 42(6), 1710–1722 (1996)
19. Watrous, J.: Zero-knowledge against quantum attacks. In: 38th Annual ACM Symposium on Theory of Computing (STOC), pp. 296–305 (2006), `http://www.cs.uwaterloo.ca/~watrous/papers.html`
20. Wehner, S., Schaffner, C., Terhal, B.M.: Cryptography from noisy storage. Physical Review Letters 100(22), 220–502 (2008)
21. Yao, A.C.-C.: Security of quantum protocols against coherent measurements. In: 27th Annual ACM Symposium on the Theory of Computing (STOC), pp. 67–75 (1995)

Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures

Jean-Sébastien Coron[1], David Naccache[2], Mehdi Tibouchi[2], and Ralf-Philipp Weinmann[1]

[1] Université du Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
{jean-sebastien.coron,ralf-philipp.weinmann}@uni.lu
[2] École normale supérieure
Département d'informatique, Groupe de Cryptographie
45, rue d'Ulm, F-75230 Paris CEDEX 05, France
{david.naccache,mehdi.tibouchi}@ens.fr

Abstract. In 1999, Coron, Naccache and Stern discovered an existential signature forgery for two popular RSA signature standards, ISO/IEC 9796-1 and 2. Following this attack ISO/IEC 9796-1 was withdrawn. ISO/IEC 9796-2 was amended by increasing the message digest to at least 160 bits. Attacking this amended version required at least 2^{61} operations.

In this paper, we exhibit algorithmic refinements allowing to attack the *amended* (currently valid) version of ISO/IEC 9796-2 for all modulus sizes. A practical forgery was computed in only two days using 19 servers on the Amazon EC2 grid for a total cost of $\simeq$ US\$800. The forgery was implemented for $e = 2$ but attacking odd exponents will not take longer. The forgery was computed for the RSA-2048 challenge modulus, whose factorization is still unknown.

The new attack blends several theoretical tools. These do not change the asymptotic complexity of Coron *et al.*'s technique but significantly accelerate it for parameter values previously considered beyond reach.

While less efficient (US\$45,000), the acceleration also extends to EMV signatures. EMV is an ISO/IEC 9796-2-compliant format with extra redundancy. Luckily, this attack does not threaten any of the 730 million EMV payment cards in circulation for *operational* reasons.

Costs are per modulus: after a first forgery for a given modulus, obtaining more forgeries is virtually immediate.

Keywords: digital signatures, forgery, RSA, public-key cryptanalysis, ISO/IEC 9796-2, EMV.

1 Introduction

RSA [34] is certainly the most popular public-key cryptosystem. A chosen-ciphertext attack against RSA textbook encryption was described by Desmedt and

Odlyzko in [17]. In RSA textbook encryption, a message m is simply encrypted as:

$$c = m^e \mod N$$

where N is the RSA modulus and e is the public exponent.

As noted in [31], Desmedt and Odlyzko's attack also applies to RSA signatures:

$$\sigma = \mu(m)^d \mod N$$

where $\mu(m)$ is an encoding function and d the private exponent. Desmedt and Odlyzko's attack only applies if the encoding function $\mu(m)$ produces integers much smaller than N. In which case, one obtains an existential forgery under a chosen-message attack. In this attack the opponent can ask for signatures of any messages of his choosing before computing, by his own means, the signature of a (possibly meaningless) message which was never signed by the legitimate owner of d.

As of today, two encoding function species co-exist:

1. *Ad-hoc encodings* are "handcrafted" to thwart certain classes of attacks. While still in use, *ad-hoc* encodings are currently being phased-out. PKCS#1 v1.5 [26], ISO/IEC 9796-1 [22] and ISO/IEC 9796-2 [23,24] are typical *ad-hoc* encoding examples.
2. *Provably secure encodings* are designed to make cryptanalysis equivalent to inverting RSA (possibly under additional assumptions such as the Random Oracle Model [2]). OAEP [3] (for encryption) and PSS [4] (for signature) are typical provably secure encoding examples.

For *ad-hoc* encodings, there is no guarantee that forging signatures is as hard as inverting RSA. And as a matter of fact, many such encodings were found to be weaker than the RSA problem. We refer the reader to [8,11,10,25,15,20] for a few characteristic examples. It is thus a practitioner's rule of thumb to use provably secure encodings whenever possible. Nonetheless, *ad-hoc* encodings continue to populate hundreds of millions of commercial products (*e.g.* EMV cards) for a variety of practical reasons. A periodic re-evaluation of such encodings is hence necessary.

ISO/IEC 9796-2 is a specific μ-function standardized by ISO [23]. In [16], Coron, Naccache and Stern discovered an attack against ISO/IEC 9796-2. Their attack is an adaptation of Desmedt and Odlyzko's cryptanalysis which could not be applied directly since in ISO/IEC 9796-2, the encoding $\mu(m)$ is almost as large as N. ISO/IEC 9796-2 can be used with hash-functions of diverse digest-sizes k_h. Coron *et al.* estimated that attacking $k_h = 128$ and $k_h = 160$ will require (respectively) 2^{54} and 2^{61} operations. After Coron *et al.*'s publication ISO/IEC 9796-2 was amended and the current official requirement (see [24]) is $k_h \geq 160$. It was shown in [13] that ISO/IEC 9796-2 can be proven secure (in the random oracle model) for $e = 2$ and if the digest size k_h is a least 2/3 the size of the modulus.

In this paper, we describe an improved attack against the currently valid (amended) version of ISO/IEC 9796-2, that is for $k_h = 160$. The new attack applies to EMV signatures as well. EMV is an ISO/IEC 9796-2-compliant format with extra redundancy. The attack is a Coron *et al.* forgery with new refinements: better message choice, Bernstein's smoothness detection algorithm (instead of trial division), large prime variant and optimized exhaustive search.

Using these refinements, a forgery for ISO/IEC 9796-2 was computed in only two days, using a few dozens of servers on the Amazon EC2 grid, for a total cost of US$800. The forgery was implemented for $e = 2$ but attacking odd exponents will not take longer. We estimate that under similar conditions an EMV signature forgery would cost US$45,000. Note that all costs are per modulus. After computing a first forgery for a given N, additional forgeries come at a negligible cost.

2 The ISO/IEC 9796-2 Standard

ISO/IEC 9796-2 is an encoding standard allowing partial or total message recovery [23,24]. Here we consider only partial message recovery. As we have already mentioned, ISO/IEC 9796-2 can be used with hash-functions $\text{HASH}(m)$ of diverse digest-sizes k_h. For the sake of simplicity we assume that k_h, the size of m and the size of N (denoted k) are all multiples of 8; this is also the case in the EMV specifications.

The ISO/IEC 9796-2 encoding function is:

$$\mu(m) = \texttt{6A}_{16} \| m[1] \| \text{HASH}(m) \| \texttt{BC}_{16}$$

where the message $m = m[1] \| m[2]$ is split in two: $m[1]$ consists of the $k - k_h - 16$ leftmost bits of m and $m[2]$ represents all the remaining bits of m. The size of $\mu(m)$ is therefore always $k - 1$ bits.

The original version of the standard recommended $128 \leq k_h \leq 160$ for partial message recovery (see [23], §5, note 4). The new version of ISO/IEC 9796-2 [24] requires $k_h \geq 160$. The EMV specifications also use $k_h = 160$.

3 Desmedt-Odlyzko's Attack

In Desmedt and Odlyzko's attack [31] (existential forgery under a chosen-message attack), the forger asks for the signature of messages of his choice before computing, by his own means, the signature of a (possibly meaningless) message that was never signed by the legitimate owner of d. In the case of Rabin-Williams signatures (see the full version of the paper [12]), it may even happen that the attacker factors N; *i.e.* a total break.

The attack only applies if $\mu(m)$ is much smaller than N and works as follows:

1. Select a bound B and let $\mathfrak{P} = \{p_1, \ldots, p_\ell\}$ be the list of all primes smaller than B.
2. Find at least $\ell + 1$ messages m_i such that each $\mu(m_i)$ is a product of primes in $\mathfrak{P}$.

3. Express one $\mu(m_j)$ as a multiplicative combination of the other $\mu(m_i)$, by solving a linear system given by the exponent vectors of the $\mu(m_i)$ with respect to the primes in $\mathfrak{P}$.
4. Ask for the signatures of the m_i for $i \neq j$ and forge the signature of m_j.

In the following we assume that e is prime; this includes $e = 2$. We let τ be the number of messages m_i obtained at step 2. We say that an integer is B-smooth if all its prime factors are smaller than B. The integers $\mu(m_i)$ obtained at step 2 are therefore B-smooth and we can write for all messages m_i, $1 \leq i \leq \tau$:

$$\mu(m_i) = \prod_{j=1}^{\ell} p_j^{v_{i,j}} \tag{1}$$

To each $\mu(m_i)$ we associate the ℓ-dimensional vector of the exponents modulo e:

$$\boldsymbol{V}_i = (v_{i,1} \bmod e, \ldots, v_{i,\ell} \bmod e)$$

Since e is prime, the set of all ℓ-dimensional vectors modulo e forms a linear space of dimension ℓ. Therefore, if $\tau \geq \ell + 1$, one can express one vector, say $\boldsymbol{V}_\tau$, as a linear combination of the others modulo e, using Gaussian elimination, which gives for all $1 \leq j \leq \ell$:

$$\boldsymbol{V}_\tau = \boldsymbol{\Gamma} \cdot e + \sum_{i=1}^{\tau-1} \beta_i \boldsymbol{V}_i$$

for some $\boldsymbol{\Gamma} = (\gamma_1, \ldots, \gamma_\ell) \in \mathbb{Z}^\ell$. That is,

$$v_{\tau,j} = \gamma_j \cdot e + \sum_{i=1}^{\tau-1} \beta_i \cdot v_{i,j}$$

Then using (1), one obtains :

$$\mu(m_\tau) = \prod_{j=1}^{\ell} p_j^{v_{\tau,j}} = \prod_{j=1}^{\ell} p_j^{\gamma_j \cdot e + \sum\limits_{i=1}^{\tau-1} \beta_i \cdot v_{i,j}} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{j=1}^{\ell} \prod_{i=1}^{\tau-1} p_j^{v_{i,j} \cdot \beta_i}$$

$$\mu(m_\tau) = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \left(\prod_{j=1}^{\ell} p_j^{v_{i,j}} \right)^{\beta_i} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}$$

That is:

$$\mu(m_\tau) = \delta^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}, \quad \text{where we denote: } \delta = \prod_{j=1}^{\ell} p_j^{\gamma_j} \tag{2}$$

Therefore, we see that $\mu(m_\tau)$ can be written as a multiplicative combination of the other $\mu(m_i)$. For RSA signatures, the attacker will ask for the signatures of $m_1, \ldots, m_{\tau-1}$ and forge the signature of m_τ using the relation:

$$\sigma_\tau = \mu(m_\tau)^d = \delta \cdot \prod_{i=1}^{\tau-1} \left(\mu(m_i)^d\right)^{\beta_i} = \delta \cdot \prod_{i=1}^{\tau-1} \sigma_i^{\beta_i} \mod N$$

In the full version of the paper [12] we describe the corresponding forgery for Rabin-Williams signatures, where, in some cases, the attacker may even factor N.

The attack's complexity depends on ℓ and on the probability that the integers $\mu(m_i)$ are B-smooth. The reader is referred to the full version of the paper [12] for a complexity analysis (see also [14]). In practice, the attack is feasible only if the $\mu(m_i)$ are relatively small (*e.g.* less than 200 bits).

4 Coron-Naccache-Stern's Attack

In ISO/IEC 9796-2, the encoding function's output $\mu(m)$ is as long as N. This thwarts Desmedt and Odlyzko's attack. Coron-Naccache-Stern's workaround [16] consisted in generating messages m_i such that a *linear combination* t_i of $\mu(m_i)$ and N is much smaller than N. Then, the attack can be applied to the integers t_i instead of $\mu(m_i)$.

More precisely, Coron *et al.* observed that it is sufficient to find a constant a and messages m_i such that:

$$t_i = a \cdot \mu(m_i) \mod N$$

is small, instead of requiring that $\mu(m_i)$ is small. Namely, the factor a can be easily dealt with by regarding $a^{-1} \mod N$ as an "additional factor" in $\mu(m_i)$; to that end we only need to add one more column in the matrix considered in Section 3. In their attack Coron *et al.* used $a = 2^8$.

Obtaining a small $a \cdot \mu(m) \bmod N$ is done in [16] as follows. From the definition of ISO/IEC 9796-2:

$$\begin{aligned} \mu(m) &= \mathtt{6A}_{16} \quad \| \; m[1] \quad \| \; \textsc{hash}(m) \quad \| \; \mathtt{BC}_{16} \\ &= \mathtt{6A}_{16} \cdot 2^{k-8} + m[1] \cdot 2^{k_h+8} + \textsc{hash}(m) \cdot 2^8 + \mathtt{BC}_{16} \end{aligned}$$

Euclidean division by N provides b and $0 \le r < N < 2^k$ such that:

$$(\mathtt{6A}_{16} + 1) \cdot 2^k = b \cdot N + r$$

Denoting $N' = b \cdot N$ one can write:

$$\begin{aligned} N' &= \mathtt{6A}_{16} \cdot 2^k + \quad (2^k - r) \\ &= \mathtt{6A}_{16} \quad \| \; N'[1] \| N'[0] \end{aligned}$$

where N' is $k + 7$ bits long and $N'[1]$ is $k - k_h - 16$ bits long.

Consider the linear combination:

$$\begin{aligned} t &= b \cdot N - a \quad \cdot \mu(m) \\ &= N' \quad - 2^8 \cdot \mu(m) \end{aligned}$$

By setting $m[1] = N'[1]$ we get:

$$\begin{aligned} t = &\quad \mathtt{6A}_{16} \,\|\, N'[1] \,\|\, N'[0] \\ &- \mathtt{6A}_{16} \,\|\, m[1] \,\|\, \textsc{hash}(m) \| \mathtt{BC00}_{16} \\ = &\quad \cancel{\mathtt{6A}_{16}} \,\|\, \cancel{N'[1]} \,\|\, N'[0] \\ &- \cancel{\mathtt{6A}_{16}} \,\|\, \cancel{N'[1]} \,\|\, \textsc{hash}(m) \| \mathtt{BC00}_{16} \\ = &\qquad N'[0] - (\textsc{hash}(m) \| \mathtt{BC00}_{16}) < 2^{k_h+16} \end{aligned}$$

For $k_h = 160$, the integer t is therefore at most 176-bits long.

The forger can thus modify $m[2]$ (and therefore $\textsc{hash}(m)$), until he gets a set of messages whose t-values are B-smooth and express one such $\mu(m_\tau)$ as a multiplicative combination of the others. As per the analysis in [16], attacking the instances $k_h = 128$ and $k_h = 160$ requires (respectively) 2^{54} and 2^{61} operations.

5 The New Attack's Building-Blocks

We improve the above complexities by using four new ideas: we accelerate Desmedt-Odlyzko's process using Bernstein's smoothness detection algorithm [6], instead of trial division; we also use the large prime variant [1]; moreover, we modify Coron *et al.*'s attack by selecting better messages and by optimizing exhaustive search to equilibrate complexities. In this section we present these new building-blocks.

5.1 Bernstein's Smoothness Detection Algorithm

Bernstein [6] describes the following algorithm for finding smooth integers.

Algorithm: Given prime numbers $p_1, \ldots, p_\ell$ in increasing order and positive integers $t_1, \ldots, t_n$, output the p_ℓ-smooth part of each t_k:

1. Compute $z \leftarrow p_1 \times \cdots \times p_\ell$ using a product tree.
2. Compute $z_1 \leftarrow z \bmod t_1, \ldots, z_n \leftarrow z \bmod t_n$ using a remainder tree.
3. For each $k \in \{1, \ldots, n\}$: Compute $y_k \leftarrow (z_k)^{2^e} \bmod t_k$ by repeated squaring, where e is the smallest non-negative integer such that $2^{2^e} \geq t_k$.
4. For each $k \in \{1, \ldots, n\}$: output $\gcd(t_k, y_k)$.

We refer the reader to [5] for a description of the product and remainder trees.

Theorem 1 (Bernstein). *The algorithm computes the p_ℓ-smooth part of each integer t_k, in $\mathcal{O}(b \log^2 b \log\log b)$ time, where b is the number of input bits.*

In other words, given a list of n_t integers $t_i < 2^a$ and the list of the first ℓ primes, the algorithm will detect the B-smooth t_i's, where $B = p_\ell$, in complexity:

$$\mathcal{O}(b \cdot \log^2 b \cdot \log \log b)$$

where $b = n_t \cdot a + \ell \cdot \log_2 \ell$ is the total number of input bits.

When n_t is very large, it becomes more efficient to run the algorithm k times, on batches of $n'_t = n_t/k$ integers. We explain in the full version of the paper [12] how to select the optimal n'_t, and derive the corresponding running time.

Bernstein recommends a number of speed-up ideas of which we used a few. In our experiments we used the *scaled remainder tree* [7], which replaces most division steps in the remainder tree by multiplications. This algorithm is fastest when FFT multiplications are done modulo numbers of the form $2^\alpha - 1$: we used this *Mersenne* FFT *multiplication* as well, as implemented in Gaudry, Kruppa and Zimmermann's GMP patch [19]. Other optimizations included computing the product z only once, and treating the prime 2 separately.

Bernstein's algorithm was actually the main source of the attack's improvement. It proved $\simeq$ 1000 faster than the trial division used in [16].

5.2 The Large Prime Variant

An integer is *semi-smooth* with respect to y and z if its greatest prime factor is $\leq y$ and all other factors are $\leq z$. Bach and Peralta [1] define the function $\sigma(u, v)$, which plays for semi-smoothness the role played by Dickman's ρ function for smoothness (see the full version of the paper [12]): $\sigma(u, v)$ is the asymptotic probability that an integer n is semi-smooth with respect to $n^{1/v}$ and $n^{1/u}$.

After an integer t_i has had all its factors smaller than B stripped-off, if the remaining factor ω is lesser than B^2 then ω must be prime. This is very easy to detect using Bernstein's algorithm. As Bernstein computes the B-smooth part z_i of each t_i, it only remains to check whether t_i/z_i is small enough. In most cases it isn't even necessary to perform the actual division since comparing the sizes of t_i and z_i suffices to rule out most non-semi-smooth numbers.

Hence, one can use a second bound B_2 such that $B < B_2 < B^2$ and keep the t_i's whose remaining factor ω is $\leq B_2$, hoping to find a second t_i with the same remaining factor ω to divide ω out. We refer the reader to the full version of the paper [12] for a detailed analysis of the large prime variant in our context.

5.3 Constructing Smaller $a \cdot \mu(m) - b \cdot N$ Candidates

In this paragraph we show how to construct smaller $t_i = a \cdot \mu(m_i) - b \cdot N$ values for ISO/IEC 9796-2. Smaller t_i-values increase smoothness probability and hence accelerate the forgery process.

We write:

$$\mu(x, h) = \mathtt{6A}_{\mathtt{16}} \cdot 2^{k-8} + x \cdot 2^{k_h+8} + h \cdot 2^8 + \mathtt{BC}_{\mathtt{16}}$$

where $x = m[1]$ and $h = \text{HASH}(m)$, with $0 < x < 2^{k-k_h-16}$.

We first determine $a, b > 0$ such that the following two conditions hold:

$$0 < b \cdot N - a \cdot \mu(0,0) < a \cdot 2^{k-8} \tag{3}$$
$$b \cdot N - a \cdot \mu(0,0) = 0 \mod 2^8 \tag{4}$$

and a is of minimal size. Then by Euclidean division we compute x and r such that:

$$b \cdot N - a \cdot \mu(0,0) = (a \cdot 2^{k_h+8}) \cdot x + r$$

where $0 \le r < a \cdot 2^{k_h+8}$ and using (3) we have $0 \le x < 2^{k-k_h-16}$ as required. This gives:

$$b \cdot N - a \cdot \mu(x,0) = b \cdot N - a \cdot \mu(0,0) - a \cdot x \cdot 2^{k_h+8} = r$$

Moreover as per (4) we must have $r = 0 \mod 2^8$; denoting $r' = r/2^8$ we obtain:

$$b \cdot N - a \cdot \mu(x,h) = r - a \cdot h \cdot 2^8 = 2^8 \cdot (r' - a \cdot h)$$

where $0 \le r' < a \cdot 2^{k_h}$. We then look for smooth values of $r' - a \cdot h$, whose size is at most k_h plus the size of a.

If a and b are both 8-bit integers, this gives 16 bits of freedom to satisfy both conditions (3) and (4); heuristically each of the two conditions is satisfied with probability $\simeq 2^{-8}$; therefore, we can expect to find such an $\{a, b\}$ pair. For example, for the RSA-2048 challenge, we found $\{a, b\}$ to be $\{625, 332\}$; therefore, for RSA-2048 and $k_h = 160$, the integer to be smooth is 170-bits long (instead of 176-bits in Coron *et al.*'s original attack). This decreases further the attack's complexity.

6 Attacking ISO/IEC 9796-2

We combined all the building-blocks listed in the previous section to compute an actual forgery for ISO/IEC 9796-2, with the RSA-2048 challenge modulus. The implementation replaced Coron *et al.*'s trial division by Bernstein's algorithm, replaced Coron *et al.*'s $a \cdot \mu(m) - b \cdot N$ values by the shorter t_i's introduced in paragraph 5.3 and took advantage of the large prime variant. Additional speed-up was obtained by exhaustive searching for particular digest values. Code was written in C++ and run on 19 Linux-based machines on the Amazon EC2 grid. The final linear algebra step was performed on a single PC.

6.1 The Amazon Grid

Amazon.com Inc. offers virtualized computer instances for rent on a pay by the hour basis, which we found convenient to run our computations. Various models are available, of which the best-suited for CPU-intensive tasks, as we write these lines, features 8 Intel Xeon 64-bit cores clocked at 2.4GHz supporting the Core2 instruction set and offering 7GB RAM and 1.5TB disk space. Renting such a capacity costs US\$0.80 per hour (plus tax). One can run up to 20 such instances in parallel, and possibly more subject to approval by Amazon (20 were enough for our purpose so we didn't apply for more).

When an instance on the grid is launched, it starts up from a disk image containing a customizable UNIX operating system. In the experiment, we ran a first instance using the basic Fedora installation provided by default, installed necessary tools and libraries, compiled our own programs and made a disk image containing our code, to launch subsequent instances with. When an instance terminates, its disk space is freed, making it necessary to save results to some permanent storage means. We simply `rsync`'ed results to a machine of ours. Note that Amazon also charges for network bandwidth but data transmission costs were negligible in our case.

All in all, we used about 1,100 instance running hours (including setup and tweaks) during a little more than two days. While we found the service to be rather reliable, one instance failed halfway through the computation, and its intermediate results were lost.

6.2 The Experiment: Outline, Details and Results

The attack can be broken down into the following elementary steps, which we shall review in turn:

1. Determining the constants $a, b, x, \mu(x, 0)$ for the RSA-2048 challenge modulus N.
2. Computing the product of the first ℓ primes, for a suitable choice of ℓ.
3. Computing the integers $t_i = bN - a\mu(m_i)$, and hence the SHA-1 digests, for sufficiently many messages m_i.
4. Finding the smooth and semi-smooth integers amongst the t_i's.
5. Factoring the smooth integers, as well as the colliding pairs of semi-smooth integers, obtaining the sparse, singular matrix of exponents, with ℓ rows and more than ℓ columns.
6. Reducing this matrix modulo $e = 2$, with possible changes in the first row (corresponding to the prime 2) depending on the Jacobi symbols $(2|t_i)$ and $(2|a)$.
7. Finding nontrivial vectors in the kernel of this reduced matrix and inferring a forgery.

Steps 2–4 were executed on the Amazon EC2 grid, whereas all other steps were run on one offline PC. Steps 3–4, and to a much lesser extent step 7, were the only steps that claimed a significant amount of CPU time.

Determining the constants. The attack's complexity doesn't depend on the choice of N. Since N has to be congruent to 5 mod 8 for Rabin-Williams signatures, we used the RSA-2048 challenge. The resulting constants were computed in SAGE [35]. We found the smallest $\{a, b\}$ pair to be $\{625, 332\}$, and the $\mu(x, 0)$ value given in the full version of the paper [12]. The integers $t_i = bN - a\mu(x, h_i)$ are thus at most 170-bits long.

Product of the first primes. The optimal choice of ℓ for 170 bits is about 2^{21}. Since the Amazon instances are memory-constrained (less than 1GB of RAM per core), we preferred to use $\ell = 2^{20}$. This choice had the additional advantage of making the final linear algebra step faster, which is convenient since this step was run on a single off-line PC. Computing the product of primes itself was done once and for all in a matter of seconds using MPIR.

Hashing. Since the attack's smoothness detection part works on batches of t_i's (in our cases, we chose batches of 2^{19} integers), we had to compute digests of messages m_i in batches as well. The messages themselves are 2048-bit long, *i.e.* as long as N, and comply with the structure indicated in the full version of the paper [12]: a constant 246-byte prefix followed by a 10-byte seed. The first two bytes identify a family of messages examined on a single core of one Amazon instance, and the remaining eight bytes are explored by increments of 1 starting from 0.

Messages were hashed using OpenSSL's SHA-1 implementation. For each message, we only need to compute one SHA-1 block, since the first three 64-byte blocks are fixed. This computation is relatively fast compared to Bernstein's algorithm, so we have a bit of leeway for exhaustive search. We can compute a large number of digests, keeping the ones likely to give rise to a smooth t_i. We did this by selecting digests for which the resulting t_i would have many zeroes as leading and trailing bits.

More precisely, we looked for a particular bit pattern at the beginning and at the end of each digest h_i, such that finding n matching bits results in n null bits at the beginning and at the end of t_i. The probability of finding n matching bits when we add the number of matches at the beginning and at the end is $(1+n/2)\cdot 2^{-n}$, so we expect to compute $2^n/(1+n/2)$ digests per selected message. We found $n = 8$ to be optimal: on average, we need *circa* 50 digests to find a match, and the resulting t_i is at most $170 - 8 = 162$ bit long (once powers of 2 are factored out).

Note that faster (*e.g.* hardware-enhanced) ways to obtain digests can significantly reduce the attack's complexity (*cf.* the full version of the paper [12]). We considered for example an FPGA-based solution called COPACOBANA [32], which could in principle perform a larger amount of exhaustive search, and accelerate the attack dramatically. It turned out that our attack was fast enough, hence pursuing the hardware-assisted search idea further proved unnecessary, but a practical attack on EMV (*cf.* section 8) could certainly benefit from hardware acceleration.

Finding smooth and semi-smooth integers. Once a batch of 2^{19} appropriate t_i's is generated, we factor out powers of 2, and feed the resulting odd numbers into our C++ implementation of Bernstein's algorithm. This implementation uses the MPIR multi-precision arithmetic library [21], which we chose over vanilla GMP because of a number of speed improvements, including J.W. Martin's patch for the Core2 architecture. We further applied Gaudry, Kruppa and Zimmermann's FFT patch, mainly for their implementation of *Mersenne* FFT multiplication, which is useful in the scaled remainder tree [7].

We looked for B-smooth as well as for (B, B_2)-semi-smooth t_i's, where $B = 16{,}290{,}047$ is the 2^{20}-th prime, and $B_2 = 2^{27}$. Each batch took $\simeq 40$ seconds to generate and to process, and consumed about 500MB of memory. We ran 8 such processes in parallel on each instance to take advantage of the 8 cores, and 19 instances simultaneously.

The entire experiment can be summarized as follows:

$$\begin{tikzcd}
 & \begin{array}{c}\text{16,230,259,553,940}\\ \text{digest computations}\end{array} \arrow[d] & \\
 & \begin{array}{c}\text{339,686,719,488 } t_i\text{'s in}\\ \text{647,901 batches of } 2^{19} \text{ each}\end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c}\text{684,365}\\ \text{smooth } t_i\text{'s}\end{array} \arrow[dr] & & \begin{array}{c}\text{366,302 collisions between}\\ \text{2,786,327 semi-smooth } t_i\text{'s}\end{array} \arrow[dl] \\
 & \text{1,050,667-column matrix} \arrow[d] & \\
 & \begin{array}{c}\text{algebra on 839,908 columns}\\ \text{having } > 1 \text{ nonzero entry}\end{array} \arrow[d] & \\
 & \text{124 kernel vectors} \arrow[d] & \\
 & \text{forgery involving 432,903 signatures} &
\end{tikzcd}$$

Finding the 1,050,667 columns (slightly in excess of the $\ell = 2^{20} = 1{,}048{,}576$ required) took a little over 2 days.

Factoring and finding collisions. The output of the previous stage is a large set of text files containing the smooth and semi-smooth t_i's together with the corresponding message numbers. Turning this data into a matrix suitable for the linear algebra stage mostly involved text manipulation in Perl to convert it to commands that could be piped into PARI/GP [33]. The resulting $1{,}048{,}576 \times 1{,}050{,}667$ matrix had 14,215,602 non-zero entries (13.5 per column on average, or 10^{-5} sparsity; the columns derived from the large prime variant tend to have twice as many non-zero entries, of course).

Linear algebra. We found non-zero kernel elements of the final sparse matrix over GF(2) using Coppersmith's block Wiedemann algorithm [9] implemented in WLSS2 [27,30], with parameters $m = n = 4$ and $\kappa = 2$. The whole computation took 16 hours on one 2.7GHz personal computer, with the first (and longest) part of the computation using 2 cores, and the final part using 4 cores.

The program discovered 124 kernel vectors with Hamming weights ranging from 337,458 to 339,641. Since columns obtained from pairs of semi-smooth numbers account for two signatures each, the number of signature queries required to produce the 124 corresponding forgeries is slightly larger, and ranges between 432,903 and 435,859.

Being written with the quadratic sieve in mind, the block Wiedemann algorithm in WLSS2 works over GF(2). There exist, however, other implementations for different finite fields.

Evidencing forgery. An interesting question is that of exhibiting a *compact evidence of forgery*. In the full version of the paper [12] we exhibit statistical evidence that a multiplicative relation between ISO/IEC 9796-2 signatures, (*i.e.* a forgery) was indeed constructed.

Fewer signature queries. In the full version of the paper [12] we address the question of reducing the number of signature queries in the attack.

7 Cost Estimates

The experiment described in the previous section can be used as a benchmark to estimate the attack's cost as a function of the size of the t_i's, denoted a; this will be useful for analyzing the security of the EMV specifications, where a is bigger (204 bits instead of 170 bits).

Table 1. Bernstein+Large prime variant. Estimated parameter trade-offs, running times and costs, for various t_i sizes.

$a = \log_2 t_i$	$\log_2 \ell$	Estimated TotalTime	$\log_2 \tau$	EC2 cost (US$)
64	11	15 seconds	20	negligible
128	19	4 days	33	10
160	21	6 months	38	470
170	22	1.8 years	40	1,620
176	23	3.8 years	41	3,300
204	25	95 years	45	84,000
232	27	19 centuries	49	1,700,000
256	30	320 centuries	52	20,000,000

Results are summarized in Table 1. We assume that the t_i's are uniformly distributed a-bit integers and express costs as a function of a. Cost figures do not include the linear algebra step whose computational requirements are very low compared to the smoothness detection step. Another difference with our experiment is that here we do not assume any exhaustive search on the t_i's; this is why the cost estimate for $a = 170$ in Table 1 is about the double of the cost of our experimental ISO/IEC 9796-2 forgery.

Running times are given for a single 2.4GHz PC. Costs correspond to the Amazon EC2 grid, as in the previous section. Estimates show that the attack is feasible up to $\simeq 200$ bits, but becomes infeasible for larger values of a. We also estimate $\log_2 \tau$, where τ is the number of messages in the forgery.

8 Application to EMV Signatures

EMV is a collection of industry specifications for the inter-operation of payment cards, POS terminals and ATMs. The EMV specifications [18] rely on ISO/IEC 9796-2 signatures to certify public-keys and to authenticate data. For instance, when an Issuer provides application data to a Card, this data must be signed using the Issuer's private key S_{I}. The corresponding public-key P_{I} must be signed by a Certification Authority (CA) whose public-key is denoted P_{CA}. The signature algorithm is RSA with $e = 3$ or $e = 2^{16} + 1$. The bit length of all moduli is always a multiple of 8.

EMV uses special message formats; 7 different formats are used, depending on the message type. We first describe one of these formats: the *Static Data Authentication, Issuer Public-key Data* (SDA-IPKD), and adapt our attack to it. We then consider the other six formats.

8.1 EMV Static Data Authentication, Issuer Public-key Data (SDA-IPKD)

We refer the reader to §5.1, Table 2, page 41 in EMV [18]. SDA-IPKD is used by the CA to sign the issuer's public-key P_{I}. The message to be signed is as follows:

$$m = \texttt{02}_{16} \| X \| Y \| N_{\text{I}} \| \texttt{03}_{16}$$

where X represents 6 bytes that can be controlled by the adversary and Y represents 7 bytes that cannot be controlled. N_{I} is the Issuer's modulus to be certified. More precisely, $X = \text{ID} \| \text{DATE}$ where ID is the issuer identifier (4 bytes) and DATE is the *Certificate Expiration Date* (2 bytes); we assume that both can be controlled by the adversary. $Y = \text{CSN} \| C$ where CSN is the 3-bytes *Certificate Serial Number* assigned by the CA and C is a constant. Finally, the modulus to be certified N_{I} can also be controlled by the adversary.

With ISO/IEC 9796-2 encoding, this gives:

$$\mu(m) = \texttt{6A02}_{16} \| X \| Y \| N_{\text{I},1} \| \text{HASH}(m) \| \texttt{BC}_{16}$$

where $N_{\text{I}} = N_{\text{I},1} \| N_{\text{I},2}$ and the size of $N_{\text{I},1}$ is $k - k_h - 128$ bits. k denotes the modulus size and $k_h = 160$ as in ISO/IEC 9796-2.

8.2 Attacking SDA-IPKD

To attack SDA-IPKD write:

$$\mu(X, N_{\text{I},1}, h) = \texttt{6A02}_{16} \cdot 2^{k_1} + X \cdot 2^{k_2} + Y \cdot 2^{k_3} + N_{\text{I},1} \cdot 2^{k_4} + h$$

where Y is constant and $h = \text{HASH}(m) \| \texttt{BC}_{16}$. We have:

$$\begin{cases} k_1 = k - 16 \\ k_2 = k_1 - 48 = k - 64 \\ k_3 = k_2 - 56 = k - 120 \\ k_4 = k_h + 8 = 168 \end{cases}$$

Generate a random k_a-bit integer a, where $36 \leq k_a \leq 72$, and consider the equation:

$$b \cdot N - a \cdot \mu(X,0,0) = b \cdot N - a \cdot X \cdot 2^{k_2} - a \cdot (\texttt{6A02}_{16} \cdot 2^{k_1} + Y \cdot 2^{k_3})$$

If we can find integers X and b such that $0 \leq X < 2^{48}$ and:

$$0 \leq b \cdot N - a \cdot \mu(X,0,0) < a \cdot 2^{k_3} \tag{5}$$

then as previously we can compute $N_{\text{I},1}$ by Euclidean division:

$$b \cdot N - a \cdot \mu(X,0,0) = (a \cdot 2^{k_4}) \cdot N_{\text{I},1} + r \tag{6}$$

where $0 \leq N_{\text{I},1} < 2^{k_3-k_4}$ as required, and the resulting $b \cdot N - a \cdot \mu(X, N_{\text{I},1}, h)$ value will be small for all values of h.

In the above we assumed Y to be a constant. Actually the first 3 bytes of Y encode the CSN assigned by the CA, and may be different for each new certificate (see the full version of the paper [12]). However if the attacker can predict the CSN, then he can compute a different a for every Y and adapt the attack by factoring a into a product of small primes.

Finding small X and b so as to minimize the value of

$$|b \cdot N - a \cdot X \cdot 2^{k_2} - a \cdot (\texttt{6A02}_{16} \cdot 2^{k_1} + Y \cdot 2^{k_3})|$$

is a Closest Vector Problem (CVP) in a bi-dimensional lattice; a problem that can be easily solved using the LLL algorithm [28]. We first determine heuristically the minimal size that can be expected; we describe the LLL attack in the full version of the paper [12].

Since $a \cdot \texttt{6A02}_{16} \cdot 2^{k_1}$ is an $(k + k_a)$-bit integer, with $X \simeq 2^{48}$ and $b \simeq 2^{k_a}$, odds are heuristically reasonable to find X and b such that:

$$0 \leq b \cdot N - a \cdot \mu(X,0,0) < 2^{(k+k_a)-48-k_a} = 2^{k-48} \simeq a \cdot 2^{k-48-k_a} = a \cdot 2^{k_3+72-k_a}$$

which is $(72 - k_a)$-bit too long compared to condition (5). Therefore, by exhaustive search we will need to examine roughly 2^{72-k_a} different integers a to find a pair (b, X) that satisfies (5); since a is k_a-bits long, this can be done only if $72 - k_a \leq k_a$, which gives $k_a \geq 36$. For $k_a = 36$ we have to exhaust the 2^{36} possible values of a.

Once this is done we obtain from (6):

$$t = b \cdot N - a \cdot \mu(X, N_{\text{I},1}, h) = r - a \cdot h$$

with $0 \leq r < a \cdot 2^{k_4}$. This implies that the final size of t values is $168 + k_a$ bits. For $k_a = 36$ this gives 204 bits (instead of 170 bits for plain ISO/IEC 9796-2). The attack's complexity will hence be higher than for plain ISO/IEC 9796-2.

In the full version of the paper [12] we exhibit concrete (a, b, X) values for $k_a = 52$ and for the RSA-2048 challenge; this required $\simeq 2^{23}$ trials (109 minutes on a single PC). We estimate that for $k_a = 36$ this computation will take roughly 13 years on a single PC, or equivalently US\$11,000 using the EC2 grid.

Table 1 shows that attacking 204-bit t_i's would cost $\simeq$ US\$84,000. As for the ISO/IEC 9796-2 attack, we can decrease this cost by first doing exhaustive search on the bits of HASH(m) to obtain a smaller t-value. We found that with 8 bits of exhaustive search cost drops to $\simeq$ US\$45,000 (without the matrix step, but in our attack algebra takes a relatively small amount of time).

8.3 Summary

In the full version of the paper [12] we provide an analysis of the other formats in the EMV specifications, with corresponding attacks when such attacks exist. We summarize results in Table 2 where an X represents a string that can be controlled by the adversary, while Y cannot be controlled. The size of the final t-value to be smooth is given in bits. Note that cost estimates are cheaper than Table 1 because we first perform exhaustive search on 8 bits of HASH(m) = SHA-1(m); however here we do take into account the cost of computing these SHA-1(m) values.

Table 2. Various EMV message formats. X denotes a data field controllable by the adversary. Y is not controllable. Data sizes for X, Y and t are expressed in bits.

EMV mode	Format	$\|X\|$	$\|Y\|$	$\|t\|$	EC2 cost (US\$)
SDA-IPKD	$02_{16}\|\|X\|\|Y\|\|N_I\|\|03_{16}$	48	56	204	45,000
SDA-SAD	Y	-	$k-176$	-	-
ODDA-IPKD	$02_{16}\|\|X\|\|Y\|\|N_I\|\|03_{16}$	48	56	204	45,000
ODDA-ICC-PKD	$04_{16}\|\|X\|\|Y\|\|N_{ICC}\|\|03_{16}\|\|\mathtt{data}$	96	56	204	45,000
ODDA-DAD1	Y	-	$k-176$	-	-
ODDA-DAD2	Y	-	$k-176$	-	-
ICC-PIN	$04_{16}\|\|X\|\|Y\|\|N_{ICC}\|\|03_{16}$	96	56	204	45,000

Table 2 shows that only four of the EMV formats can be attacked, with the same complexity as the SDA-IPKD format. The other formats seem out of reach because the non-controllable part Y is too large.

Note that these attacks do not threaten any of the 730 million EMV payment cards in use worldwide for *operational* reasons: the Issuer and the CA will never accept to sign the chosen messages necessary for conducting the attack.

9 Conclusion

This paper exhibited a *practically exploitable* flaw in the *currently valid* ISO/IEC 9796-2 standard and a conceptual flaw in EMV signatures. The authors recommend the definite withdrawal of the *ad-hoc* encoding mode in ISO/IEC 9796-2 and its replacement by a provably secure encoding function such as PSS.

References

1. Bach, E., Peralta, R.: Asymptotic semismoothness probabilities. Mathematics of Computation 65(216), 1701–1715 (1996)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of CCS 1993, pp. 62–73. ACM, New York (1993)
3. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption: How to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
4. Bellare, M., Rogaway, P.: The Exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
5. Bernstein, D.J.: Fast Multiplications and its applications. Algorithmic Number Theory 44 (2008)
6. Bernstein, D.J.: How to find smooth parts of integers (2004/05/10), `http://cr.yp.to/papers.html#smoothparts`
7. Bernstein, D.J.: Scaled remainder trees (2004/08/20), `http://cr.yp.to/papers.html#scaledmod`
8. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
9. Coppersmith, D.: Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. Mathematics of Computation 62(205), 333–350 (1994)
10. Coppersmith, D., Coron, J.-S., Grieu, F., Halevi, S., Jutla, C.S., Naccache, D., Stern, J.P.: Cryptanalysis of ISO/IEC 9796-1. Journal of Cryptology 21, 27–51 (2008)
11. Coppersmith, D., Halevi, S., Jutla, C.: ISO 9796-1 and the new, forgery strategy, Research contribution to P.1363 (1999), `grouper.ieee.org/groups/1363/Research`
12. Coron, J.S., Naccache, D., Tibouchi, M., Weinmann, R.P.: Practical Cryptanalysis of ISO / IEC 9796-2 and EMV Signatures, Cryptology ePrint Archive, Report 2009/203, `http://eprint.iacr.org/`
13. Coron, J.-S.: Security proofs for partial domain hash signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 613–626. Springer, Heidelberg (2002)
14. Coron, J.-S., Desmedt, Y., Naccache, D., Odlyzko, A., Stern, J.P.: Index calculation attacks on RSA signature and encryption. Index calculation attacks on RSA signature and encryption Designs, Codes and Cryptography 38(1), 41–53 (2006)
15. Coron, J.-S., Naccache, D., Joye, M., Paillier, P.: New attacks on PKCS#1 v1.5 encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 369–381. Springer, Heidelberg (2000)
16. Coron, J.-S., Naccache, D., Stern, J.P.: On the security of RSA padding. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 1–18. Springer, Heidelberg (1999)
17. Desmedt, Y., Odlyzko, A.: A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 516–522. Springer, Heidelberg (1986)
18. EMV, Integrated circuit card specifications for payment systems, Book 2. Security and Key Management. Version 4.2 (June 2008), `http://www.emvco.com`
19. Gaudry, P., Kruppa, A., Zimmermann, P.: A gmp-based implementation of Schőnhage-Strassen's large integer multiplication algorithm. In: Proceedings of ISSAC 2007, Waterloo, Ontario, Canada, pp. 167–174. ACM Press, New York (2007)

20. Grieu, F.: A chosen messages attack on the ISO/IEC 9796-1 signature scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 70–80. Springer, Heidelberg (2000)
21. Hart, W.B., et al.: Multiple Precision Integers and Rationals, `http://www.mpir.org`
22. ISO / IEC 9796, Information technology – Security techniques – Digital signature scheme giving message recovery, Part 1: Mechanisms using redundancy (1999)
23. ISO / IEC 9796-2, Information technology – Security techniques – Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function (1997)
24. ISO / IEC 9796-2:2002, Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms (2002)
25. Joux, A., Naccache, D., Thomé, E.: When e-th roots become easier than factoring. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 13–28. Springer, Heidelberg (2007)
26. Kaliski, B.: pkcs#1: RSA Encryption Standard, Version 1.5, RSA Laboratories (November 1993)
27. Kaltofen, E., Lobo, A.: Distributed matrix-free solution of large sparse linear systems over finite fields. Algorithmica 24, 331–348 (1999)
28. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 513–534 (1982)
29. Lenstra Jr., H.: Factoring integers with elliptic curves. Annals of Mathematics 126(2), 649–673 (1987)
30. Lobo, A.: WLSS2: an implementation of the homogeneous block Wiedemann algorithm, `www4.ncsu.edu/~kaltofen/software/wiliss`
31. Misarsky, J.-F.: How (not) to design RSA signature schemes. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 14–28. Springer, Heidelberg (1998)
32. Paar, C., Schimmer, M.: COPACOBANA: A Codebreaker for DES and other ciphers, `www.copacobana.org`
33. The PARI Group, PARI/GP, version 2.3.4, Bordeaux (2008), `http://pari.math.u-bordeaux.fr`
34. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM 21, 120–126 (1978)
35. The sage development team, sage mathematics software, Version 3.3 (2009), `http://www.sagemath.org`

How Risky Is the Random-Oracle Model?

Gaëtan Leurent[1] and Phong Q. Nguyen[2]

[1] DGA and ENS, France
http://www.eleves.ens.fr/leurent/
[2] INRIA and ENS, France
http://www.di.ens.fr/~pnguyen/

Abstract. RSA-FDH and many other schemes secure in the Random-Oracle Model (ROM) require a hash function with output size larger than standard sizes. We show that the random-oracle instantiations proposed in the literature for such cases are weaker than a random oracle, including the proposals by Bellare and Rogaway from 1993 and 1996, and the ones implicit in IEEE P1363 and PKCS standards: for instance, we obtain a 2^{67} preimage attack on BR93 for 1024-bit digests. Next, we study the security impact of hash function defects for ROM signatures. As an extreme case, we note that any hash collision would suffice to disclose the master key in the ID-based cryptosystem by Boneh *et al.* from FOCS '07, and the secret key in the Rabin-Williams signature for which Bernstein proved tight security at EUROCRYPT '08. Interestingly, for both of these schemes, a slight modification can prevent these attacks, while preserving the ROM security result. We give evidence that in the case of RSA and Rabin/Rabin-Williams, an appropriate PSS padding is more robust than all other paddings known.

1 Introduction

The Random-Oracle Model (ROM), in which hash functions are viewed as random oracles, goes back to at least Fiat and Shamir [17]. Popularized by Bellare and Rogaway [1], it has been the subject of much debate. On the one hand, it is widespread in research papers and standards, because ROM schemes are usually more efficient, their security proofs can be based on well-studied assumptions, and can sometimes be tight. In fact, many standardized public-key schemes are, at best, only proven secure in the ROM, *e.g.* RSA-OAEP [2] and RSA-PSS [3].

On the other hand, the ROM is not an assumption (on the hash function): there is no random-oracle definition which a public function can hope to satisfy. A security proof in the standard model (SM) precisely gives assumptions which are sufficient to ensure security properties. By contrast, a ROM security proof only shows that an attack which does not break the proof assumptions must exploit a property not satisfied by the random-oracle simulation of the security proof. This allowed Canetti *et al.* [10] to build signature and encryption schemes which are secure in the ROM, but insecure for any (efficient) implementation of the random oracle, because such implementations can be simulated by a Universal Turing machine, and therefore be distinguished [33] from a random oracle. However, the constructions [4, 10, 20, 33] showing the limitations of the ROM are arguably "unnatural" and significantly differ from real-world constructions. Still, one should be careful with idealized security models: like all Merkle-Damgård/Davies-Meyer hash functions, MD5 was provably collision-resistant [7, 50] (up to the birthday bound) in the ideal cipher model (with respect to the block cipher underlying the compression function); yet, computing MD5 collisions only costs a few seconds now [43].

This stresses the importance of studying the actual security of schemes proven in the ROM. Unfortunately, very few ROM schemes have also been proven secure in the SM [8, 42]; and for several cases, there is evidence that a security proof in the SM is unlikely [15, 28, 34, 39]. Recent breakthroughs in the cryptanalysis of hash functions [45, 47, 48] have shown that standard hash functions like MD5 or SHA-1 are far from behaving like random oracles. However, the impact on the public-key world has been limited so far, with the exception of [45], which constructs two colliding X.509 certificates for different identities and public keys, and has recently been extended in [43] to construct a rogue CA certificate.

But to study the actual security, one needs to know how the random oracle will be instantiated in practice, should the scheme ever be used. Often, the random-oracle output size matches that of standard hash functions (like 160 bits for SHA-1) or the upcoming SHA-3. In this case, standard hash functions are most likely to be used, despite well-known properties of MD-iterated hash functions (such as the derivation of $h(m_1||m_2)$ from $h(m_1)$ and m_2) which make them easily differentiable from random oracles. But RSA-FDH [3] and many other ROM schemes (such as [6, 9, 12, 13, 19, 24, 27]) actually require a "non-standard" hash function. First, the output may not be a uniformally distributed bitstring: it could be residue classes, or elliptic curve points, *etc.*, fortunately it is known how to deal with such situations given an instantiation with arbitrary output $\{0,1\}^n$. But if the output bit-length is larger than standard sizes (*e.g.* RSA-FDH which needs at least 1024 bits), it is unclear how the oracle will be instantiated. To the best of our knowledge, the only proposals of random-oracle instantiations supporting arbitrary outbit bit-length are the following: two historical instantiations proposed by Bellare and Rogaway in their seminal papers [1] (on the ROM) and [3] (on FDH and PSS), recent constructions by Coron *et al.* in the full version of [14], and the instantiations implicit in PKCS#1 v2.1 [41] and IEEE P1363 [23] standards. It seems that none of these instantiations have been analyzed in the literature, except [14] in the indifferentiability framework of Maurer *et al.* [33].

This also raises the question of the impact of potential defects in random-oracle instantiations. When an article provides a ROM security proof, it usually does not say how to instantiate the random oracle, neither what might happen if the hash function does not behave like a random oracle. Assume that Alice implements a scheme ROM-secure under a well-known computational assumption. Several years later, the assumption still stands, but Alice learns that her random-oracle implementation is not as perfect as she thought: Should Alice worry? Are the risks confined to chosen-message existential forgery and ciphertext distinguishability? If Alice had the choice between two (equally efficient) schemes secure in the ROM under the same assumption, maybe Alice would rather choose the least risky one, in terms of robustness to hash function defects: the scheme with less stringent security requirements on the hash function, and the least affected if ever the requirements are not satisfied.

Our Results. We analyze the main proposals [1, 3, 23, 41] of random-oracle instantiations supporting arbitrary output bit-length. While none of these proposals made it clear what was exactly the expected security guarantee, one might argue that an instantiation with output bit-length n should offer $2^{n/2}$ resistance to collisions, and 2^n resistance to preimages, as required for SHA-3. We show that the proposals fall short of those bounds: for instance, for 1024-bit digests, we give a 2^{67} preimage attack on BR93 [1] and a 2^{106} collision attack on BR96 [3]. We note that the instantiations implicit in PKCS [41] and IEEE [23] standards are not collision-resistant: independently of the output size, collisions follow directly from SHA-1 collisions, which cost only 2^{61} [26, 47]. And we show that when applied to the compression functions of MD5 or SHA-1, the theoretical constructions of Coron *et al.* [14] are no more collision-resistant than MD5 or SHA-1 themselves. This highlights the difficulty of instantiating/simulating a random oracle, and motivates the study of the impact of hash defects on schemes secure in the ROM.

As a second contribution, we show that, while the ROM is useful to detect structural flaws, it can hide very different security requirements on the hash function, and very different damages in case of hash defects, independently of the computational assumption and the tightness of the security reduction. We illustrate this phenomenon with a well-known class of ROM schemes: padding-based signatures from trapdoor one-way functions, such as RSA, Rabin, Rabin-Williams and ESIGN. While it is often believed that a hash collision may at worst give rise to an existential forgery, we show that for several secure signatures proposed in the literature [3, 6, 9, 19, 24], collisions or slight hash function defects can have much more dramatic consequences, namely key-recovery attacks. However, this does not contradict the ROM security proofs, and does not mean that such signatures cannot be secure with a proper hash function. Our most interesting examples are related to Rabin and Rabin-Williams signatures, but the issues are not restricted to factoring-based schemes: in the full version, we show similar problems for a recent lattice-based signature scheme [19]. For instance, we remark that any hash collision discloses the master key in the ID-based cryptosystem of Boneh *et al.* [9], and the secret key in the Rabin-Williams signature scheme for which Bernstein [6] recently proved tight security, which was not mentioned in either [6, 9]. Interestingly, we show that a slight modification of the signing process can prevent our collision-based key-recovery attacks, while preserving the ROM security result. We give evidence that in the case of RSA and Rabin/Rabin-Williams, an appropriate PSS padding (with large salt) is more robust than all other paddings known, especially deterministic ones and randomized paddings with small randomness, including Katz-Wang [27] which achieve tightness.

To put things into perspective, consider the Rabin-Williams signature included in the IEEE P1363 standard [23]: it can be optionally deterministic or randomized, but the deterministic version has no tight reduction. Since tightness was a key factor in standardizing RSA-PSS over RSA-FDH, one might be tempted to replace this deterministic Rabin-Williams by its "tight" variant analyzed by Bernstein [6], but doing so would have led to a chosen-message key-recovery attack from any SHA-1 collision because such collisions provide collisions on the RO-instantiation of the P1363 standard. Alternatively, there would have also been a chosen-message key-recovery attack if IEEE P1363 had simply used the BR93 RO-instantiation [1], due to the preimage attack.

Some of the problems we discuss are related to the issue of how to derandomize a signature scheme. As a side-remark, we show that the derandomization technique proposed by Granboulan [22] to fix ESIGN during the NESSIE European project is not completely sound: when applied to ESIGN or DSA, it may leak the secret key. Finally, while this paper focuses on the ROM, we believe that in general, not just in the ROM, it is also interesting to identify necessary (possibly minimal) assumptions for security, and to assess the security impact when assumptions do not hold. This is useful when comparing cryptographic schemes, and can complement provable security results.

Road map. We assume the reader is familiar with hash functions, the ROM [1] and provable security for signatures [21]. In Sect. 2, we recall and analyze random-oracle instantiations for large output size. Next, we study the implications of hash function defects for ROM signatures based on a trapdoor one-way function and a padding: we recall such schemes in Sect. 3. In Sect. 4, we study the robustness of derandomized Rabin/Rabin-Williams signatures, which are used in [6] and the ID-based cryptosystem of [9]. In Sect. 5, we compare the robustness of RSA and Rabin/Rabin-Williams.

2 Random-Oracle Instantiations for Large Output

In this section, we describe and analyze random-oracle instantiations supporting arbitrary output bit-length that have been proposed in the literature, namely [1, 3, 14] and the instantiations implicit in the PKCS#1 v2.1 [41] and IEEE P1363 [23] standards. For completeness, we also briefly discuss collision-resistant hash functions such as [11, 32]. While some of the instantiations make use of MD5, that alone is insufficient to discard them. Indeed, though the collision-resistance of MD5 is seriously broken [29, 48], many usages of MD5 are not threatened yet: for instance, there is still no practical attack on HMAC-MD5.

2.1 The 1993 Instantiation by Bellare and Rogaway

Description. In their seminal paper on the ROM [1], Bellare and Rogaway gave guidelines to instantiate a random oracle (see [1, Sect. 6]), but the only explicit construction is the following one, which we call BR93:

- Let $h_4 : \{0,1\}^{512} \to \{0,1\}^{128}$ be the first compression function of MD5, that is the compression function evaluated with the initial IV of MD5.
- Let $h' : \{0,1\}^{256} \to \{0,1\}^{64}$ defined by $h'(x)$ being the first 64 bits of $h_4((xx) \oplus C)$, for a randomly chosen 512-bit constant C. The function h' defines a pseudo-random number generator $h''(x) : \{0,1\}^{192} \to \{0,1\}^*$ by counter as follows[1]: $h''(x) = h'(x\langle 0\rangle)||h'(x\langle 1\rangle)||h'(x\langle 2\rangle)\dots$ where $\langle i\rangle$ is the encoding of i into 64 bits.
- Finally, the BR93 instantiation of the random oracle is the truncation (prefix) of $h(x) : \{0,1\}^* \to \{0,1\}^*$ defined as follows. First, one applies a padding to x by adding a bit 1 and enough bits 0 to obtain a bitstring x' whose bit-length is a multiple of 128. Then, if we divide x' into 128-bit blocks as $x' = x'_0 \dots x'_{n-1}$, then $h(x) = h''(x'_0\langle 0\rangle) \oplus h''(x'_1\langle 1\rangle) \oplus \cdots \oplus h''(x'_{n-1}\langle n-1\rangle)$, that is, $h(x)$ is the XOR of the n streams produced by each of the x'_i.

Weaknesses. We claim that BR93 is much weaker than a random oracle with respect to collision and preimage resistance, independently of the choice of the underlying compression function (MD5 here): the main idea is to adapt Wagner's generalized birthday algorithm [46]. Concretely, we give:

- a collision attack on $(k+2)r$ bits with complexity $2^k \cdot r2^r$ using messages of 2^k blocks. For instance, a collision attack on 1024 bits with messages of 2^{30} blocks costs $2^{30} \cdot 32 \cdot 2^{32} = 2^{67}$ elementary operations. If we limit the message size to 2^{14} blocks, the complexity is $2^{14} \cdot 64 \cdot 2^{64} = 2^{84}$. The complexity does not depend on the size of the underlying compression function.
- a preimage attack on $(k+1)r$ bits with complexity of $2^k \cdot r2^r$ using messages of 2^k blocks. For instance, a preimage attack on 1024 bits with messages of 2^{31} blocks costs $2^{31} \cdot 32 \cdot 2^{32} = 2^{68}$ elementary operations.

Generalized birthday. Recall Wagner's generalized birthday algorithm [46]. The basic operation is the general join $\bowtie_j$: $L \bowtie_j L'$ consists of all elements of $L \times L'$ such that their j least significant bits match:

$$L \bowtie_j L' = \left\{ l \oplus l' : (l,l') \in L \times L' \;\middle|\; (l \oplus l')^{[0..j-1]} = 0^j \right\}.$$

Assume that we are given several lists $L_0, L_1, \dots$, each of size 2^r. Our goal is to find $l_0 \in L_0, l_1 \in L_1, \dots$ such that $\bigoplus l_i = 0$. The idea is to join the lists using a binary tree. We build the first level with $L_{01} = L_0 \bowtie_r L_1$, $L_{23} = L_2 \bowtie_r L_3$, and so on. By the birthday paradox, these new lists should still contain about 2^r elements. On the next level, we build $L_{0123} = L_{01} \bowtie_{2r} L_{23}$. Since the elements of L_{01} and L_{23} already agree on their r lower bits, we are only doing a birthday paradox on the bits r to $2r$, so we still expect to find 2^r elements. If we start with 2^k lists, on the last level we end up with one list of 2^r elements which all begin with kr zeros. In this list, we expect to find two elements that agree on $2r$ extra bits, so that we have a collision on $(k+2)r$ bits.

Collisions. We now use Wagner's algorithm to find collisions on BR93. For each message block x'_i, we will consider 2^r possible values, and build a list L_i with the

[1] The paper [1] actually says 224 instead of 192 for the input size of h', but that would be incompatible with the definition of h' as $224 + 64 = 288 > 256$.

resulting $h''(x'_i\langle i\rangle)$. Then we can use Wagner's attack on these lists. A collision attack on $(k+2)r$ bits will have a complexity of $2^k \cdot r2^r$ using messages of 2^k blocks. For instance, a collision attack on 1024 bits with messages of 2^{30} blocks costs $2^{30} \cdot 32 \cdot 2^{32} = 2^{67}$ elementary operations. If we limit the message size to 2^{14} blocks, the complexity is $2^{14} \cdot 64 \cdot 2^{64} = 2^{84}$. Note that this complexity does not depend on the size of the underlying hash function.

Preimages. We can also use Wagner's algorithm to find preimages on BR93. If we want a preimage of H, we first replace L_0 by $L_0 \oplus H$. On the last level of the tree, we will still have one list of 2^r elements which all begins with kr zeros, but instead of looking for a collision on $2r$ extra bits, we look for an element with r extra zeroes. This element corresponds to a message x such that $H \oplus h(x) = H \oplus h''(x'_0\langle 0\rangle) \oplus h''(x'_1\langle 1\rangle) \oplus ...h''(x'_{2^k-1}\langle 2^k-1\rangle) =_{(k+1)r} 0$, *i.e.* the $(k+1)r$ first bits of $h(x)$ agree with H. A preimage attack on $(k+1)r$ bits will have a complexity of $2^k \cdot r2^r$ using messages of 2^k blocks. A preimage attack on 1024 bits with messages of 2^{31} blocks costs $2^{31} \cdot 32 \cdot 2^{32} = 2^{68}$ elementary operations.

2.2 The 1996 Instantiation by Bellare and Rogaway

Description. In their paper [3] on PSS, Bellare and Rogaway proposed another instantiation [3, App. A], which we call BR96: let H = MD5 or SHA-1, and define $h_{BR96}(x)$ as the appropriate truncation (prefix) of:

$$H(\texttt{const}\langle 0\rangle x)||H(\texttt{const}\langle 1\rangle x)||H(\texttt{const}\langle 2\rangle x)||\ldots$$

where the constant `const` should be unique to h. If another instantiation is needed, one should change `const`. BR96 is very close to a previous construction described in the OAEP paper [2] where H above is replaced by the 80-bit truncation of SHA-1, but that construction was not explicitly recommended to instantiate a random oracle, though it was used as a building block in an implementation of RSA-OAEP.

Weaknesses. We note that since H is a MD function, h_{BR96} can be viewed as the concatenation of MD functions, where each $H_i : x \mapsto H(\texttt{const}\langle i\rangle x)$ is a distinct iterative hash function, which implies:

– h_{BR96} can be distinguished from a random oracle. More precisely, BR96 suffers from the same extension problems as any MD function: if the output size of h_{BR96} is an exact multiple of that of H, and if m_1 has appropriate size, then $h_{BR96}(m_1||m_2)$ can be computed from $h_{BR96}(m_1)$ and m_2.

– h_{BR96} is weaker than a random oracle with respect to collision and preimage resistance, independently of the choice of the underlying hash function (except its output size), thanks to Joux's multicollision technique [25].

Recall that Joux [25] showed that the concatenation of two or more MD-iterated hash functions has roughly the same security as a single hash function. More precisely, if one concatenates k iterated hash functions with an internal size of n bits each, [25] finds a collision in the concatenation for a workload of $n^{k-1} \times 2^{n/2}$, and preimages for an imprecise workload of $\mathsf{poly}(n^k)2^n$.

A closer analysis of the collision attack shows that the cost $n^{k-1} \times 2^{n/2}$ can actually be reduced to $[(n/2)^{k-1} + (n/2)^{k-2} + \cdots + 1] \times 2^{n/2} \leq (n/2)^{k-1} \times (n/2)/(n/2-1) \times 2^{n/2} \approx (n/2)^{k-1} \times 2^{n/2}$. And there seems to be a more efficient preimage attack by generalizing the basic preimage attack against two hash functions as follows. First, build a $2^{n^{k-1}/2^{k-2}}$-multicollision on the first hash function F_1, and look for an extra block that maps this multicollision to the target value of F_1. Then build a multicollision in F_2 using the messages of the first multicollision: each collision in F_2 requires a set of $2^{n/2}$ messages, which will be built from $n/2$ colliding pairs in F_1. Thus we should get a $2^{n^{k-2}}/2^{k-3}$-multicollision in F_1. We will also use the last n colliding pairs for a preimage search on F_1. This gives us a $2^{n^{k-2}}/2^{k-3}$-multicollision in $F_1||F_2$ which is also a preimage. We apply the technique iteratively to build a 2^n-multicollision for $F_1||F_2||...F_{k-1}$ which is also a preimage. If we compute F_k on the set of 2^n colliding messages, we expect to find one preimage against the full concatenation. The most expensive steps of this attack are the preimage search, because the collision finding steps all have complexity $O(n^k \times 2^{n/2})$. The preimage step on F_i requires to compute F_i on 2^n messages, which are made of n block pairs of length $n^{i-2}/2^{i-2}$ and one block of length $n^{i-2}/2i-3$. If we do an amortized analysis, each computation requires to hash 2 blocks from message pairs, and the final block, which gives a cost of $n^{i-2}/2^{i-4} \times 2^n$. The cost of the full preimage search is roughly equivalent to the cost of the last preimage search, which is $n^{k-2}/2^{k-4} \times 2^n$.

We now apply this to BR96. For instance, if H is MD5, we can find collisions in 1024 bits of the output with a workload of essentially $64^7 \cdot 2^{64} = 2^{106}$, where the colliding messages will be of length $64^7 = 2^{42}$ blocks; and we can find preimages of 1024 bits of the output with a workload of $128^6/2^4 \cdot 2^{128} = 2^{166}$. These complexities are clearly impractical and do not threaten [3], but they are much lower than the theoretical security of a 1024-bit random oracle.

For the same reason, BR96 is also malleable. For instance, we can create pairs of messages x_0, x_1 such that $H(\texttt{const}\langle i\rangle x_0) = H(\texttt{const}\langle i\rangle x_1)$ for all i's except the last one. We will build a multicollision set of $2^{n/4}$ such messages, and we expect to find one quadruplet such that $H(x_0) \oplus H(x_1) \oplus H(x_2) \oplus H(x_3) = 0$. In the full version, we show how this kind of malleability can be exploited to attack GPV [19].

As another example, consider the previous instantiation of BR96 for 1024-bit digests, using MD5. We can find two near-collisions where the most significant 384 bits are equal, with of a workload of essentially $64^2 \cdot 2^{64} = 2^{76}$. Such near-collisions give existential forgeries for the historical version [37] of ESIGN.

2.3 Recent Instantiations by Coron *et al.* (CDMP)

Description. Coron *et al.* [14] (CDMP) proposed several variations of Merkle-Damgård to build a random oracle from an (ideal) compression function or an (ideal) block-cipher using the Davies-Meyer mode. They proposed four variants of MD for *input* domain extensions (namely, *Prefix-Free Encoding*, *Dropping*

Some Output Bits, *Using NMAC*, and *Using HMAC*) and one scheme (only in the full version of [14]) for *output* domain extension. The output extension scheme is similar to BR96, but the counter is included after the message (which is reminiscent of the MGF1 pseudo-random number generator used in several standards [23, 41]):

$$h_{CDMP}(x) = H(x\langle 0\rangle)||H(x\langle 1\rangle)||H(x\langle 2\rangle)||\dots$$

where H is one of the four input extension schemes. This choice is due to efficiency considerations, but we will see that it has a strong security impact. The main advantage of [14] is its security proof: all the constructions are proved indifferentiable from a random oracle (in the sense of Maurer *et al.* [33]), if the underlying compression function is a random oracle, or if it uses the Davies-Meyer mode with an ideal block cipher. However, no recommendation is given in [14] for the choice of the underlying compression function (or the underlying block cipher for Davies-Meyer). So strictly speaking, unlike [1, 3], there was no fully concrete proposal of a random-oracle instantiation: still, one may want to apply the constructions to usual compression functions.

Weaknesses. One should be careful not to overestimate the significance of indifferentiability security proofs: in practice, there is no ideal compression function. It was shown by [5] that none of the CDMP constructions necessarily preserve collision-resistance: they give (theoretical) examples of collision-resistant compression functions for which the resulting hash function is not collision-resistant.

While [14] was presented as a fix to the MD construction, we show that if one applies these fixes to MD5 or SHA-1, one can still find collisions in the new hash function (independently of the chosen output length) with the same cost as the original MD5 or SHA-1. This means that [14] does not address collision attacks on MD5 and SHA-1. To see this, we first show that the four input extensions are not collision resistant if applied to the compression functions of MD5 or SHA-1. This is trivial for *Dropping Some Output Bits*, *Using NMAC*, and *Using HMAC*, because these constructions are nested: an inner collision becomes an outer collision. So the only potentially tricky case is *Prefix-Free Encoding*, for which [14] proposed only two instantiations:

– prepend the message size as the first block. It turns out that MD5/SHA-1 collision attacks [47, 48] can be extended to this case, because the number of blocks of colliding messages produced is equal and already known in advance, and it is well-known that existing MD5/SHA-1 collision attacks can be extended to any given IV.
– use the first bit of each message block as a flag to distinguish the last message block. Since the number of blocks in MD5/SHA-1 colliding messages is very small, and the first bit of each block is random looking, we can simply produce random collisions until one has the required form.

Now, because of the iterated structure of the four input extensions, these collisions give rise to collisions in the output extension h_{CDMP}. More generally, while h_{CDMP} is indifferentiable from a random oracle if H is also indifferentiable, any collision in H becomes a collision in h_{CDMP} if H has an iterative structure like MD or the four input extensions: namely, $H(x_0) = H(x_1)$ implies $H(x_0\langle i\rangle) = H(x_1\langle i\rangle)$ and therefore $h_{CDMP}(x_0) = h_{CDMP}(x_1)$.

Hence, we have shown that if the CDMP constructions are applied to the compression functions of MD5 or SHA-1 for an arbitrary output size, the cost of producing collisions remains essentially the same as for MD5 or SHA-1 [26]. Of course, one could try to use different compression functions, but no concrete recommendation is given in [14].

2.4 Instantiations in PKCS and IEEE Standards

Description. No cryptographic standard currently specifies a random-oracle instantiation for arbitrary size. However, several instantiations are implicit in PKCS #1 v2.1 [41] and IEEE P1363 [23], because RSA-OAEP [2] and RSA-PSS [3] are standardized:

– RSA-OAEP requires two random oracles G and H with small input size (less than the RSA modulus), which are both instantiated in PKCS by the MGF1 pseudo-random number generator [41]. Recall that MGF1 is simply a hash function in counter mode like h_{CDMP}, except that the counter is over four bytes: $\mathrm{MGF1}(x) = h(x\langle 0\rangle)||h(x\langle 1\rangle)||h(x\langle 2\rangle)||\ldots$, where h is either SHA-1 or a SHA-2.
– RSA-PSS also requires two random oracles G and H, but while G still has small input size, H has a small output size but possibly large inputs. In PKCS, H is instantiated by SHA-1 or SHA-2, and G is instantiated by MGF1.

Thus, none of the oracles required by RSA-OAEP and RSA-PSS have both a large input and output as would be required by RSA-FDH. Still, MGF1 is a potential random-oracle instantiation, because it supports arbitrarily large input and output.

There is another implicit instantiation in IEEE P1363 [23]. Indeed, it includes a Rabin-Williams signature using a variant of the PSS encoding [3] (as described in [24]) called EMSA-PSS in [41] and EMSA4 in [23]: the main difference between EMSA-PSS and PSS [3] (described in Sect. 3.2) is that the message is first hashed before going through the PSS encoding. But it is specified in [23] that the salt can optionally be set to zero, in which case "*the signature scheme is deterministic, similar to Full-Domain Hashing*". Thus, one can view EMSA-PSS with zero salt as an instantiation of a FDH: since the padding constants are zero, this amounts to essentially hash the message twice in a row, then apply MGF1; concatenate the output and the input of MGF1, and append the "BC" byte.

Weaknesses. The case of MGF1 has already been analyzed with the CDMP case in the previous subsection: using SHA-1 or any MD-iterated hash function, the cost of producing collisions in MGF1 remains as low as for the underlying hash function. And EMSA-PSS with zero salt is clearly no more collision-resistant than the underlying hash function. Note also that the "BC" byte makes it differentiable from a random oracle. Hence, independently of the output size chosen, finding collisions on the PKCS/IEEE instantiations costs as low as for the underlying hash function MD5 or SHA-1.

2.5 Provably Collision-Resistant Hash Functions

To conclude this section, we briefly discuss the case of hash functions which are provably collision-resistant under appropriate computational assumptions. Though not designed nor recommended to instantiate random oracles, they might be potential candidates since they usually support large output size. But it is folklore that none should be viewed nor used as a random oracle, because they have special properties which are not satisfied by a random oracle, typically malleability. Consider for instance two recent collision-resistant hash functions:

- VSH [11], which is collision-resistant provided that a certain problem related to factorization is hard. The output set is $\mathbb{Z}_N^\times$, where N is hard to factor.
- SWIFFT [32], which is (asymptotically) collision-resistant and one-way, provided that certain lattice approximation problems are hard. The smallest output size is 528 bits, but larger sizes are possible.

These functions are malleable in the following sense. In [32], it is noted that for any two inputs x_1 and x_2 such that $x_1 + x_2$ is a valid input, $\text{SWIFFT}(x_1) + \text{SWIFFT}(x_2) = \text{SWIFFT}(x_1 + x_2)$. By definition of VSH [11], it is easy to generate $M_0 \neq M_1$ such that $4\text{VSH}(M_0) \equiv \text{VSH}(M_1) \pmod N$ where N is the public modulus. More generally, for any product $s > 1$ of very small distinct primes (chosen among the primes used by the VSH compression function), it is easy to generate $M_0 \neq M_1$ such that $s^2\text{VSH}(M_0) \equiv \text{VSH}(M_1) \pmod N$.

We will see that such malleability relationships can be exploited to attack certain signature schemes. The malleability of SWIFFT can be exploited to attack the GPV signature [19] (see the full version), and the malleability of VSH can be exploited to attack Rabin and Rabin-Williams signatures. But we stress that neither VSH or SWIFFT were recommended to be used with these signatures.

3 Padding-Based Signatures in the Random-Oracle Model

We study the impact of hash function defects for the class of ROM-secure signatures obtained by combining a trapdoor one-way function/permutation and a padding. More precisely, we consider secure versions of RSA, Rabin, Rabin-Williams and ESIGN with appropriate paddings: FDH [1], PSS [3], PFDH [12] and KW [27]. This section briefly recalls these components.

3.1 Signatures from Trapdoor One-Way Functions

Let σ denote the raw signature algorithm which, given as input a message $m \in \mathcal{M}$ outputs a signature $\sigma(m) \in \mathcal{S}$: the algorithm can be either deterministic or probabilistic. We consider signatures based on a trapdoor one-way function $f : \mathcal{S} \to \mathcal{M}$:

- If f is 1-to-1, we obtain a deterministic scheme with $\sigma(m) = f^{-1}(m)$.
- If f is many-to-1, we obtain a probabilistic scheme with $\sigma(m)$ selected uniformly at random in the set $f^{-}(m)$ of preimages: we require that the trapdoor enables such preimage sampling.

Verification checks that a given signature s belongs to $\mathcal{S}$ and that $f(s) = m$.
RSA. Let (N, e, d) be the usual RSA keys where the exponent d is secret. We have $\mathcal{M} = \mathcal{S} = \mathbb{Z}_N$, and take the RSA trapdoor permutation $f(x) = x^e$ defined over $\mathcal{M}$, whose inverse is $f^{-1}(x) = x^d$.
Rabin [40]. Let $N = pq$ be a usual RSA modulus. Then we take the squaring function $f(x) = x^2$, which is a 4-to-1 mapping from $\mathcal{S} = \mathbb{Z}_N^\times$ to the subgroup $\mathcal{M}$ of quadratic residues mod N. Inverting f is equivalent to factoring N.
Rabin-Williams [49]. This is a variation of Rabin signatures based on tweaks, using a modulus $N = pq$ such that $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$. This has two notable features: one can take $\mathcal{M} = \mathbb{Z}_N$ (rather than having to deal with quadratic residues), and one can obtain mappings onto $\mathcal{M}$ which are either 4-to-1 or 1-to-1, and whose inversion is equivalent to factoring. For any $m \in \mathcal{M}$, there are exactly four triplets $(e, f, s) \in \mathcal{S} = \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, N-1\}$ such that $m \equiv efs^2$, and these so-called tweaked square roots can all be efficiently computed using p and q. Furthermore, the *principal* tweaked square root is the only square root such that e is 1 if m is a square modulo q, otherwise -1; f is 1 if em is a square modulo p, otherwise 2; and s is a square modulo $N = pq$. We thus obtain two trapdoor one-way functions, depending on the choice of $\mathcal{S}$:

- By taking the 4-to-1 mapping, we obtain the probabilistic signature scheme PRW.
- By taking the 1-to-1 mapping where $\mathcal{S}$ is confined to principal tweaked square roots, we obtain the deterministic signature scheme DRW. Ignoring technical details, this is essentially the Rabin-Williams used in IEEE P1363 [23].

ESIGN [36, 37]. We only give an informal description. ESIGN uses an RSA modulus of the form $N = p^2q$ such that p, q have bit-length k, and N has bit-length $3k$. There is a small public exponent $e \geq 8$ not necessarily coprime with $\phi(N)$. The one-way function is the truncation of the RSA permutation $f(x) = x^e$ to its k most significant bits. This is a many-to-one mapping from $\mathcal{S} = \mathbb{Z}_N$ to $\mathcal{M} = \{0, \ldots, \lfloor N/2^{2k} \rfloor\}$, whose inversion problem is called AER [36].

3.2 Paddings

A padding Π specifies how to sign arbitrary messages $m \in \{0,1\}^*$: it may be deterministic or randomized. Then the signature is $\sigma(\Pi(m))$, with additional data in the case of PFDH. With the exception of PSS, all the following paddings use a full-domain hash h from $\{0,1\}^*$ to $\mathcal{M}$.
FDH [1]. This deterministic padding is simply $\Pi(m) = h(m)$.
PFDH [12]. One selects a salt $r \leftarrow_{\mathcal{R}} \{0,1\}^k$, and let $\Pi(m) = h(m||r)$ where h is a random oracle from $\{0,1\}^*$ to $\mathcal{M}$. The signature must include the salt r.
KW [27]. If m has never been signed, select a one-bit salt $r \leftarrow_{\mathcal{R}} \{0,1\}$, and let $\Pi(m) = h(r||m)$.
PSS [3]. Assume to simplify that $\mathcal{M} = \{0,1\}^k$ where $k \geq k_0 + k_1$ for some integers k_0 and k_1. Two random oracles $h : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$ and $g : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k-k_1}$ are used. We let g_1 be the function which on input $w \in \{0,1\}^{k_1}$ returns the first k_0 bits of $g(w)$, and let g_2 be the function which on input $w \in \{0,1\}^{k_1}$

returns the remaining $k - k_0 - k_1$ bits of $g(w)$. For any message $m \in \{0,1\}^*$, one selects $r \in_R \{0,1\}^{k_0}$ and let $w = h(m\|r)$ and $r^* = g_1(w) \oplus r$. Finally, $\Pi(m) = w\|r^*\|g_2(w)$.
Standards. The PKCS [41] and IEEE P1363 [23] standards actually implement a slightly different version of PSS, called EMSA-PSS [24]: the main difference is that the message is first hashed before going through the PSS encoding. As mentioned earlier in Sect. 2.4, Rabin-Williams in IEEE P1363 is implemented as essentially DRW-PSS, but the salt can optionally be zero, in which case it becomes DRW-FDH with a specific RO-instantiation.

3.3 Derandomization

Several schemes [6, 9, 27] crucially require to derandomize the signature or the padding:

- The main scheme analyzed by Bernstein [6] is derandomized PRW-FDH with the requirement that if ever the same message is submitted twice, the same signature should be output.
- The ID-based cryptosystem of Boneh *et al.* [9] uses derandomized Rabin-FDH for mapping identities to secret keys.
- To implement RSA-KW, Katz and Wang [27] suggested to select the one-bit salt deterministically from a secret key and the message.

We will see that how the derandomization is performed has a big impact on the security. Bernstein [6] did not specify how this derandomization should be performed: he only mentioned that if ever the same message is submitted twice, the same signature should be output, otherwise an attacker would be able to compute two random tweaked square roots of the same element (by signing twice the same message), which discloses the secret key. But Katz and Wang discussed [27, Sect. 4.1] that issue in the context of RSA signatures, and proposed the following methods:

- KW1: select the nonce as $r = h'(K||m)$, where h' is an independent random oracle, K is an additional secret key and m is the message.
- KW2: select $r = h'(K||h(m))$ with the same notations as KW1.
- KW3: select $r = F_K(m)$ where F is a PRF and K is an additional secret key. This is the method used in the ID-based cryptosystem [9], which is shown to preserve the ROM security proof.

In order to derandomize ESIGN to fix its security proof (see [38, 44]), Granboulan [22] earlier proposed a slightly different method: select $r = \phi(h(m)||K||c)$, where ϕ is a one-way function with uniformly distributed output, K is an additional secret key, and c is an optional counter. The counter is necessary in ESIGN, because the signature process may actually fail for certain nonces. This derandomization technique was later adopted with a specific choice of ϕ in the revised submission [18] of ESIGN to the NESSIE European project.

3.4 Security Results

It is well-known that for any trapdoor permutation f, the signatures FDH, PFDH and PSS are all provably secure in the ROM under the hardness of inverting the permutation, but the security proof is loose [1, 16]. In the particular case of RSA, all these security proofs can be improved, by exploiting the multiplicativity of RSA: RSA-FDH remains loose [1, 3, 12], but RSA-PFDH and RSA-PSS have a tight reduction provided that the salt is sufficiently large (see [12, 16]). Surprisingly, RSA-KW also has a tight reduction [27]. These security results also apply to DRW (under the factoring assumption), because DRW uses a 1-to-1 mapping which is homomorphic (see [24]).

The picture is a bit different with Rabin and PRW, since they use 4-to-1 mappings, but squaring is homomorphic. The derandomized versions of Rabin-FDH and PRW-FDH have a tight reduction (see [9] for Rabin-FDH and [6] for PRW-FDH). Rabin-PSS has a tight reduction (see [3]). For a complete picture of the ROM security of all Rabin-Williams variants, see [6].

ESIGN-FDH [36] and ESIGN-PSS [30] both have a loose security proof under the AER assumption, but the proof of ESIGN-FDH requires a more restricted security model than usual (see [38, 44]). There is no tightness because the one-way function is not multiplicative.

4 Robustness of Derandomized Signatures

We now study the robustness of derandomized signatures described in Sect. 3, for the derandomization methods proposed in [9, 22, 27], which we described in Sect. 3.3. This derandomization is crucial for the tightness of certain security proofs. We focus on derandomized Rabin-FDH and PRW-FDH.

4.1 Soundness

First of all, one should make sure that the derandomization technique does not affect the security proof of the randomized scheme. For KW3, this was proved in [9, App. B] using of course the PRF assumption. And a similar argument can be proved for KW1 and KW2, but both require another random oracle, which is debatable if the goal is to understand what are the minimal assumptions.

For the fourth randomization method however, Granboulan [22] only gave an informal argument. In fact, we note that his method is not completely sound. More precisely, if the informal argument was correct, it would also apply to the choice $r = \phi(m||K||c)$. Now, assume that we take for ϕ an MD-iterated function for which it is easy to find collisions, but still, the function is one-way with uniformly distributed output: one potential example is SHA-1. Then an adversary could create a collision (m, m') where m and m' have the same length, which would imply that $\phi(m||K||c) = \phi(m'||K||c)$ for all K and c. This means that by querying the signature of m and m', the adversary would obtain two pairs message-signature which both used the same nonce r: in the case of ESIGN, this discloses the secret factorization of the modulus, and in the case of DSA, this clearly discloses the secret key. This means that [22] did not give the right assumption: instead of a one-way function with uniform output, one should consider a PRF such as in KW3.

4.2 Robustness to Collisions

We now look at the security if the full-domain hash h has defects, namely collisions. We show that any hash collision suffices to disclose the master key in the ID-based cryptosystem of Boneh *et al.* [9], and the secret key in the Rabin-Williams signature scheme for which Bernstein proved tight security [6], because of the way derandomization is performed. But we also show that these attacks can be prevented by slightly modifying the derandomization.

The idea is, of course, to obtain two random preimages of a known element $m \in \mathcal{M}$. Doing so for the trapdoor one-way function f of Rabin and PRW discloses the secret factorization of the modulus with probability 1/2. Let $h : \{0,1\}^* \rightarrow \mathcal{M}$ be the random-oracle instantiation, to be paired with derandomized versions of Rabin or PRW. We have the following chosen-message key-recovery attack:

– Assume that the attacker is able to generate a collision (M_0, M_1) on h. Then $H(M_0) = H(M_1)$ with $M_0 \neq M_1$.
– The attacker queries the signing oracle on M_0 and M_1, and obtains the signature s_0 and s_1.
– Depending on how the derandomization is performed, we claim that s_0 and s_1 will be two random preimages of the same element $h(M_0) = h(M_1) \in \mathbb{Z}_N$, in which case it is easy to obtain the factorization of N with probability 1/2.

  This is true in either of the following cases:

  • If one follows the informal method of Bernstein [6]: since M_0 and M_1 are different, the signer is not required to output the same preimage, so each preimage will be chosen at random.
  • If one follows KW1 [27], because $h(M_0) = h(M_1)$ is independent from $h'(K\|M_0) = h'(K\|M_1)$ where K denotes the secret key and h' is another random oracle.
  • If one follows KW3 [27] as in the ID-based cryptosystem [9], because $h(M_0) = h(M_1)$ is independent from $F_K(M_0) = F_K(M_1)$.

  But if one follows KW2 [27], the same preimage will be output, and the attack will fail. An alternative method to prevent the attack is to use the following variant of KW3: $r = F_K(h(m))$ so that collisions on h gives collisions on r. For both variants, the previous key-recovery attacks no longer work, but that does not mean that the schemes are immune to collisions: any hash collision gives rise to an existential forgery. The interest of KW2 and the KW3 variant is that they both decrease (but not remove) the security impact of collisions.

Independently of the random-oracle instantiation and the choice of the derandomization, these weaknesses also appear if one considers fault attacks. Indeed, a similar attack works if the adversary is able to submit twice the same message, and perturbate the calculation of the nonce, using fault attacks.

By contrast, the deterministic version DRW-FDH (and the one implemented in IEEE P1363 [23]) is immune to such attacks. It might help to see a concrete example. Assume that we plug the compression function of MD5 into the CDMP random-oracle construction [14] (see Sect. 2.4), and that we use this instantiation as a full-domain hash for DRW-FDH and derandomized PRW-FDH. Then, because of the indifferentiability framework [33], both signature schemes become provably secure under the factoring assumption, in the ideal cipher model (with respect to the MD5 block cipher) or in the random oracle model (with respect to the MD5 compression function). But in practice, there is an instant chosen-message key-recovery attack on the PRW scheme, which fails on the DRW one.

4.3 Robustness to Malleability

The previous key-recovery attack can be adapted to malleability variants of collisions on the hash function. To simplify, consider first the case of derandomized PRW-FDH: a similar attack works for derandomized Rabin-FDH. Assume that the attacker is able to generate a pair (M_0, M_1) of distinct messages such that:

$$4h(M_0) \equiv h(M_1) \pmod N. \tag{1}$$

From Sect. 2, we know that this is easy if ever h is VSH [11] using the same modulus N, even though it might be hard to find collisions on VSH, but we stress that it was never suggested to use VSH for Rabin/Rabin-Williams that way: we give this example to show that solving (1) is not necessarily harder than finding collisions. Solving (1) is also possible (with more effort) if h is BR93 [1]: select any M_0 then apply the preimage attack to find M_1 satisfying (1). Again, the attacker queries the signing oracle on M_0 and M_1, which gives rise to tweaked square roots $(e_i, f_i, s_i) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, N-1\}$ of $h(M_i)$. Note though that there is a one-to-one correspondance between the four tweaked square roots of $h(M_0)$ and the four tweaked square roots of $h(M_1)$, thanks to (1). More precisely, if (e, f, s) is a tweaked square root of $h(M_0)$, then $(e, f, 2s \bmod N)$ is a tweaked square root of $h(M_1)$. This implies that $(e_0, f_0, 2s_0 \bmod N)$ and (e_1, f_1, s_1) are two "independent" random tweaked square roots of $h(M_1)$, which means that one can factor N with probability 1/2.

Obviously, this attack is independent of the way derandomization is performed, and can be adapted to other malleability properties. For instance, similar attacks apply if one is able to find a pair (M_0, M_1) of distinct messages such that $h(M_0) \equiv -h(M_1) \pmod N$, or $k^2 h(M_0) \equiv h(M_1) \pmod N$ for some known $k \in \mathbb{Z}_N^\times$.

Furthermore, as opposed to collision attacks, the previous attack can be adapted to DRW-FDH. Starting again from (1), let $(e_i, f_i, s_i) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, N-1\}$ be the principal tweaked square root of $h(M_i)$. Because 4 is a square mod p and q, (1) implies that $e_0 = e_1$ and $f_0 = f_1$. Since $(e_0, f_0, 2s_0 \bmod N)$ is a tweaked square root of $h(M_1)$, we have $4s_0^2 \equiv s_1^2 (\bmod N)$, and therefore $s_1 \times (2s_0)^{-1} \bmod N$ is a square root of 1 mod N. But it must be a non-trivial square root because it has different Legendre symbols mod p and

q: indeed, both s_0 and s_1 are squares mod N, while $\left(\frac{2}{p}\right) = 1$ and $\left(\frac{2}{q}\right) = -1$. Hence, this discloses the factorization of N. This attack can be generalized if congruence (1) is replaced by $k^2 h(M_0) \equiv h(M_1) \pmod N$ for any known $k \in \mathbb{Z}_N^\times$ such that $\left(\frac{k}{p}\right) \neq \left(\frac{k}{q}\right)$, which is slightly more restrictive than the PRW case.

There are similar attacks for the Rabin case.

4.4 Robustness to Preimages

The previous attacks also show that both derandomized PRW-FDH and DRW-FDH become strongly insecure if the full-domain hash function h is not one-way, like BR93 [1]. Alternatively, one can simply select $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, N-1\}$ uniformly at random, and compute $m = efs^2 \pmod N$. By inverting h, one obtains a message M such that $m = h(M)$. Finally, by signing the message M with either DRW-FDH or derandomized PRW-FDH, one will obtain another tweaked square root of m (principal or not), which will disclose the factorization of N with probability at least 1/2 because (e, f, s) is a random tweaked square root. A similar attack works for the Rabin case.

5 Compared Robustness of Signatures

5.1 RSA Signatures

The PKCS#1 v2.1 standard [41] uses RSA-PSS since Sept. 1999 (or more precisely, the variant RSA-EMSA-PSS [24] of RSA-PSS), and it has been reported that one of the main reasons why RSA-PSS was selected over RSA-FDH was the tightness of the security proof. If tightness was the main factor, one might now be tempted to select RSA-KW over RSA-PSS, because the salt in RSA-KW is reduced to one bit (which can be deterministically derived from the secret key and the message). However, by comparing the robustness of RSA signatures with respect to potential defects in the random-oracle instantiation, a different picture emerges.

Robustness to collisions. Because RSA-FDH and RSA-EMSA-PSS are hash-and-sign schemes, they do not tolerate collisions: any collision obviously leads to a chosen-message existential forgery. Similarly, any collision leads to a chosen-message existential forgery on RSA-KW, with probability 1/2 because of the one-bit salt. One may think that the probabilistic schemes RSA-PFDH and RSA-PSS are more robust. In this direction, Numayama *et al.* [35] showed that RSA-PFDH tolerates collisions in a weakened ROM, but their model does not take into account MD-iterated hash functions. We observe that if h is a MD-iterated hash function, then any collision in h with the same number of blocks gives rise to a chosen-message existential forgery on RSA-PFDH and RSA-PSS. This is because RSA-PFDH and RSA-PSS both use $h(m||r)$. And if $h(m_1) = h(m_2)$ where m_1 and m_2 have the same number of blocks, then $h(m_1||r) = h(m_2||r)$ for any r. This implies that for both RSA-PFDH and RSA-PSS, any signature of m_1 is also valid for m_2. It can be noted that if RSA-PFDH and RSA-PSS had used $h(r||m)$ instead of $h(m||r)$, then the ROM security proofs would remain valid, but the previous attack would fail.

Robustness to preimages. It is easy to prove that if the full-domain hash is not one-way, then there are chosen-message universal forgery attacks on RSA-FDH, RSA-PFDH and RSA-KW. On the other hand, preimages in h do not seem to provide stronger attacks than chosen-message existential forgeries on RSA-PSS.

Conclusion. While RSA-KW has a much better security reduction than RSA-FDH, there are essentially the same attacks on both RSA-KW and RSA-FDH as soon as there are defects in the full-domain hash. On the other hand, RSA-PSS with a large salt seems more robust than all other paddings, especially if one uses $h(r||m)$ instead of $h(m||r)$. This differs from the conclusion of [31], where it was argued that RSA-FDH was the best method known to sign with RSA.

5.2 Rabin and Rabin-Williams

Based on Sect. 4, the advantages of Rabin over Rabin-Williams are unclear from a security point of view. There are two benefits with Rabin-Williams: one can select $\mathcal{M} = \mathbb{Z}_N$, and one can use a 1-to-1 mapping instead of a 4-to-1 mapping. This 1-to-1 mapping avoids collision attacks or the fault attacks on derandomization: This suggests that DRW is preferable to both PRW and Rabin. And for the same reason as for RSA, a PSS padding with large salt seems more robust than all other paddings. Furthermore, when using Rabin or PRW, the size of the salt is extremely important to avoid key-recovery replay attacks. By submitting many times the same message, an adversary would obtain two random preimages of the same element, hence the factorization. We illustrate this phenomenon in the full version, with a converse to the security proof of [3].

Acknowledgements. We thank Dan Boneh, Charles Bouillaguet, Louis Granboulan, Eike Kiltz, Kaoru Kurosawa, Alfred Menezes, Martijn Stam and the anonymous reviewers for their comments. Part of this work was supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676 ECRYPT II.

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS 1993, pp. 62–73. ACM Press, New York (1993)
2. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
3. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
4. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)

5. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
6. Bernstein, D.J.: Proving tight security for Rabin-Williams signatures. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 70–87. Springer, Heidelberg (2008)
7. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 320. Springer, Heidelberg (2002)
8. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: FOCS 2007, pp. 647–657. IEEE Computer Society, Los Alamitos (2007)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004); (electronic) Preliminary version at STOC 1998
11. Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an efficient and provable collision-resistant hash function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)
12. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
13. Coron, J.S.: Security proof for partial-domain hash signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 613–626. Springer, Heidelberg (2002)
14. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
15. Dodis, Y., Oliveira, R., Pietrzak, K.: On the generic insecurity of the full domain hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
16. Dodis, Y., Reyzin, L.: On the power of claw-free permutations. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 55–73. Springer, Heidelberg (2003)
17. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
18. Fujisaki, E., Kobayashi, T., Morita, H., Oguro, H., Okamoto, T., Okazaki, S.: ESIGN-D specification. Submission to the NESSIE European Project, available on the NESSIE webpage (2002)
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008. ACM, New York (2008)
20. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: FOCS 2003. IEEE Computer Society, Los Alamitos (2003)
21. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
22. Granboulan, L.: How to repair ESIGN. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 234–240. Springer, Heidelberg (2003)
23. IEEE: P1363: Standard specifications for public-key cryptography, `http://grouper.ieee.org/groups/1363/`

24. Jonsson, J.: Security proofs for the RSA-PSS signature scheme and its variants. Report 2001/053 of the Cryptology ePrint Archive (2001)
25. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
26. Joux, A., Peyrin, T.: Hash functions and the (amplified) boomerang attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
27. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: CCS 2003. ACM, New York (2003)
28. Kiltz, E., Pietrzak, K.: On the security of padding-based encryption schemes (or: Why we cannot prove OAEP secure in the standard model). In: EUROCRYPT 2009. LNCS. Springer, Heidelberg (2009)
29. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105 (2006)
30. Kobayashi, T., Fujisaki, E.: Security of ESIGN-PSS. IEICE Transactions 90-A(7), 1395–1405 (2007)
31. Koblitz, N., Menezes, A.J.: Another look at "provable security". J. Cryptology 20(1), 3–37 (2007)
32. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)
33. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
34. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 111. Springer, Heidelberg (2002)
35. Numayama, A., Isshiki, T., Tanaka, K.: Security of digital signature schemes in weakened random oracle models. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 268–287. Springer, Heidelberg (2008)
36. Okamoto, T., Fujisaki, E., Morita, H.: TSH-ESIGN: Efficient digital signature scheme using trisection size hash. Submission to IEEE P1363a (1998)
37. Okamoto, T.: A fast signature scheme based on congruential polynomial operations. IEEE Transactions on Information Theory 36(1), 47–53 (1990)
38. Okamoto, T., Stern, J.: Almost uniform density of power residues and the provable security of ESIGN. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 287–301. Springer, Heidelberg (2003)
39. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (2005)
40. Rabin, M.: Digital signatures and public key functions as intractable as factorization. Technical report, MIT Laboratory for Computer Science, TR-212 (1979)
41. RSA Laboratories: PKCS #1 v2.1: RSA cryptography standard (June 14, 2002)
42. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)
43. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: CRYPTO 2009. LNCS. Springer, Heidelberg (2009)

44. Stern, J., Pointcheval, D., Malone-Lee, J., Smart, N.P.: Flaws in applying proof methodologies to signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 93. Springer, Heidelberg (2002)
45. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
46. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
47. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
48. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
49. Williams, H.C.: A modification of the RSA public-key encryption procedure. IEEE Trans. Inform. Theory 26(6), 726–729 (1980)
50. Winternitz, R.S.: A secure one-way hash function built from DES. In: Proc. IEEE Symposium on Security and Privacy, pp. 88–90 (1984)

Abstraction in Cryptography

Ueli Maurer

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

Abstract. Abstraction means to eliminate irrelevant details from consideration, thereby focusing only on the relevant aspects of a problem or context. Abstraction is of paramount importance in most scientific fields, especially in computer science and mathematics. The purpose of abstraction is to provide, at the same time, simpler definitions, higher generality of results, simpler proofs, improved elegance, and often better didactic suitability.

Abstraction can be a gradual process and need not be unique, but in many contexts, the highest achievable level of abstraction, once identified, appears natural and stable. For example, the abstract and natural concepts of a group or a field in algebra capture exactly what is required to prove many important results.

In the spirit of algebraic abstraction, we advocate the definition and use of higher levels of abstraction in cryptography, with the goal of identifying the highest possible level at which a definition or theorem should be stated and proved. Some questions one can ask are: What are abstractions of a system, a game, indistinguishability, a hybrid argument, a reduction, indifferentiability, or of (universal) composability? What are abstractions of efficient and negligible, and at which level of abstraction can computational and information-theoretic models be unified? And, of course: Can the abstract viewpoint lead to new concepts and results that are perhaps otherwise missed?

Asymptotically Good Ideal Linear Secret Sharing with Strong Multiplication over *Any* Fixed Finite Field

Ignacio Cascudo[1], Hao Chen[2], Ronald Cramer[3], and Chaoping Xing[4]

[1] Department of Mathematics, University of Oviedo, Spain
`icascudo@orion.ciencias.uniovi.es`
[2] Software Engineering Institute, East China Normal University, Shanghai 20062, China
`haochen@sei.ecnu.edu`
[3] CWI, Amsterdam & Mathematical Institute, Leiden University, The Netherlands
`http://www.cwi.nl/~cramer`
[4] Division of Mathematical Sciences, Nanyang Technological University, Singapore
`http://www3.ntu.edu.sg/home/xingcp/`

Abstract. This work deals with "MPC-friendly" linear secret sharing schemes (LSSS), a mathematical primitive upon which secure multi-party computation (MPC) can be based and which was introduced by Cramer, Damgaard and Maurer (EUROCRYPT 2000). Chen and Cramer proposed a special class of such schemes that is constructed from algebraic geometry and that enables efficient secure multi-party computation over fixed finite fields (CRYPTO 2006). We extend this in four ways. First, we propose an abstract coding-theoretic framework in which this class of schemes and its (asymptotic) properties can be cast and analyzed. Second, we show that for *every finite field* $\mathbb{F}_q$, there exists an infinite family of LSSS over $\mathbb{F}_q$ that is asymptotically good in the following sense: the schemes are "*ideal*," i.e., each share consists of a single $\mathbb{F}_q$-element, and the schemes have *t-strong multiplication* on n players, where the corruption tolerance $\frac{3t}{n-1}$ tends to a constant $\nu(q)$ with $0 < \nu(q) < 1$ when n tends to infinity. Moreover, when $|\mathbb{F}_q|$ tends to infinity, $\nu(q)$ tends to 1, which is optimal. This leads to explicit lower bounds on $\widehat{\tau}(q)$, our measure of *asymptotic optimal corruption tolerance.* We achieve this by combining the results of Chen and Cramer with a dedicated field-descent method. In particular, in the $\mathbb{F}_2$-case there exists a family of binary t-strongly multiplicative ideal LSSS with $\frac{3t}{n-1} \approx 2.86\%$ when n tends to infinity, a one-bit secret and just a one-bit share for every player. Previously, such results were shown for $\mathbb{F}_q$ with $q \geq 49$ a square. Third, we present an infinite family of ideal schemes with t-strong multiplication that does not rely on algebraic geometry and that works over every finite field $\mathbb{F}_q$. Its corruption tolerance vanishes, yet still $\frac{3t}{n-1} = \Omega(1/(\log \log n) \log n)$. Fourth and finally, we give an improved non-asymptotic upper bound on corruption tolerance.

1 Introduction

This work deals with "MPC-friendly" linear secret sharing schemes (LSSS), an abstract mathematical primitive upon which secure multi-party computation (MPC) can be based and which was introduced by Cramer, Damgaard and Maurer [13]. Chen and Cramer [8] proposed a special class of such schemes that is constructed from algebraic geometry and that enables efficient secure multi-party computation over fixed finite fields. For every finite field $\mathbb{F}_q$ where q is a square with $q \geq 49$, they presented an infinite family of LSSS over $\mathbb{F}_q$ that is asymptotically good in the following sense. First, the schemes are "*ideal*": each share consists of a single $\mathbb{F}_q$-element. Second, the schemes have *t-strong multiplication* [8, 9, 12, 13] on n players, where the corruption tolerance $\frac{3t}{n-1}$ tends by a constant $\nu(q)$ with $0 < \nu(q) < 1$ when n tends to infinity. Moreover, when $|\mathbb{F}_q|$ tends to infinity, $\nu(q)$ tends to 1, which is optimal (since it is well-known that $3t + 1 \leq n$ always). In short, strong multiplication is a property that enables to "perfectly securely" verify multiplicative relations among secret-shared values, with *error probability equal to zero.* This is a crucial subroutine at the heart of MPC. Please refer to [13] for the details.

These schemes of [8] enjoy algebraic properties similar to those of Shamir's scheme (linearity, (strong)-multiplication), and MPC protocols in the strongest information-theoretic model [2, 3] (i.e., perfect security against a computationally unbounded threshold adversary that corrupts some fraction of the players) are quite similar (see [8, 13]). The significance of these schemes, however, derives from the fact that the number of players is not bounded by the size of the finite field as is the case for Shamir's scheme [26]. In fact, in these schemes the number of players is unbounded even if the finite field is fixed. In the corresponding MPC-protocols, the total number of field elements communicated will typically be the same, with the notable difference, however, that the field elements are now taken in a field of constant rather than linearly increasing size. This makes sense, for instance, if the function that is securely computed is defined over a small, constant size field, say $\mathbb{F}_2$. The price to be paid is only a *constant* fractional decrease compared to the corruption tolerance of (non-asymptotic) Shamir-based MPC-protocols, in which up to 1/3 of the players may be corrupted. The construction of these "MPC-friendly" schemes from [8] is based on the existence of families of algebraic curves of finite fields with a good ratio between the number of rational points and their genus and the use of their algebraic function fields. See [6, 8, 13] for a full discussion. Chen, Cramer, Goldwasser, de Haan and Vaikuntanatan [7] have shown similar results for schemes with multiplication rather than strong multiplication by a construction from arbitrary classical codes rather than algebraic geometric ones.

The results from [7, 8] have found remarkable applications in the breakthrough work of Ishai, Kushilevitz, Ostrovsky and Sahai [20] on two-party zero-knowledge for circuit-satisfiability with low communication, in that of Ishai, Prabhakaran and Sahai [19] on oblivious transfer, as well as in the work of Damgaard, Nielsen and Wichs [14] on isolated zero-knowledge. In all these cases this helped improving the communication efficiency. It is important to note that all these applications use secret sharing and MPC as abstract primitives, where players are *not actual, real-world players* but are part of *virtual* processes. Moreover, the number of these virtual players is typically large in order to make certain error-probabilities small enough or in order to approximate a certain asymptotical advantage. This has amplified the relevance of secure computation and secret sharing even further, and in particular it adds further relevance to asymptotical study of these primitives.

In this paper we extend these results in four ways. First, we propose an abstract coding-theoretic framework in which this class of schemes and its (asymptotic) properties can be cast and analyzed. Concretely, we introduce a special class of codes $\mathcal{C}^\dagger(\mathbb{F}_q)$ and a measure $\widehat{t}(C)$ on a code $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$, the corruption tolerance of C, so that when C is viewed to represent an "ideal" LSSS, $\widehat{t}(C)$ measures the maximum t for which it has t-strong multiplication. We also define the *asymptotic optimal corruption tolerance* $\widehat{\tau}(q)$ of such a class over $\mathbb{F}_q$, the main parameter for our asymptotic analysis.

Second, we show that for *every finite field* $\mathbb{F}_q$, there exists an infinite family of LSSS over $\mathbb{F}_q$ that is asymptotically good in the following sense: the schemes are "*ideal*," i.e., each share consists of a single $\mathbb{F}_q$-element, and the schemes have *t-strong multiplication* on n players, where the corruption tolerance $\frac{3t}{n-1}$ tends to a constant $\nu(q)$ with $0 < \nu(q) < 1$ when n tends to infinity. Moreover, when $|\mathbb{F}_q|$ tends to infinity, $\nu(q)$ tends to 1, which is optimal. This leads to explicit lower bounds on $\widehat{\tau}(q)$. Our method combines the algebraic geometric schemes of Chen and Cramer with our dedicated but elementary field-descent method based on "multiplication-friendly functions," which maps $C \in \mathcal{C}^\dagger(\mathbb{F}_{q^m})$ to $C' \in \mathcal{C}^\dagger(\mathbb{F}_q)$ in such a way that corruption tolerance does not degrade too much. In particular, in the $\mathbb{F}_2$-case there exists a family of binary t-strongly multiplicative ideal LSSS with $\frac{3t}{n-1} \approx 2.86\%$ when n tends to infinity, a one-bit secret and just a one-bit share for every player. Previously, such results were only shown to hold over $\mathbb{F}_q$ with $q \geq 49$ a square.

Third, we present an infinite family of ideal schemes with t-strong multiplication that does not rely on algebraic geometry and that works over every finite field $\mathbb{F}_q$. Its corruption tolerance vanishes, yet still $\frac{3t}{n-1} = \Omega(1/(\log\log n)\log n)$. Fourth and finally, we give an improved non-asymptotic upper bound on corruption tolerance.

The outline of this paper is as follows. After the preliminaries in Section 2, we revisit in Section 3 the results in [7] about the construction of LSSS from linear codes, where we focus mainly on privacy and reconstruction. In some cases we define new notions and prove stronger results needed in the sequel. In Section 4 we define $\mathcal{C}^\dagger(\mathbb{F}_q)$, $\widehat{t}(C)$ and $\widehat{\tau}(q)$, and prove some properties. In Section 5 we show that for every finite fields $\mathbb{F}_q$, the asymptotic optimal corruption tolerance can be bounded away from zero, i.e., $\widehat{\tau}(q) > 0$ for all finite field $\mathbb{F}_q$, and we give explicit lower bounds. In Section 6 we state the consequences for LSSS with strong multiplication explicitly and in Section 7 we present the elementary example with ("not-so-fast") vanishing asymptotic corruption tolerance. In Section 8, we give our non-asymptotic upper bound on corruption tolerance. In Section 9 we conclude by stating some open problems.

Finally, we note that, in upcoming work [5], we further improve our asymptotic lower bounds on optimal corruption tolerance using more advanced methods from algebraic geometry, especially for small values of q.

2 Preliminaries

2.1 Basic Coding Theory

We review some notions from basic coding theory (see e.g. [22] or [17]) that are relevant to this work and we also introduce some conventions specific to this paper. Let n be a non-negative integer and let k be an integer with $0 \leq k \leq n+1$. An $[n+1,k]_q$-code C over the finite field $\mathbb{F}_q$ is a k-dimensional subspace C of the $n+1$-dimensional $\mathbb{F}_q$-vector space $\mathbb{F}_q^{n+1}$. The length $n+1$ of such a code C is denoted $\ell(C)$. We define $n(C) = \ell(C) - 1$. If $\mathbf{c} \in C$, $(c_0, c_1, \ldots, c_n) \in \mathbb{F}_q^{n+1}$ denotes its coordinate vector. In particular, we use the set $\mathcal{I}(C) = \{0, 1, \ldots, n\}$ to index the coordinates, unless otherwise stated. A linear code over $\mathbb{F}_q$ is an $[n+1,k]_q$-code for some k, n. If $B \subset \mathcal{I}(C)$ is a non-empty set and if $\mathbf{x} = (x_0, x_1, \ldots, x_n) \in \mathbb{F}_q^{n+1}$, $\mathbf{x}_B$ denotes the vector $(x_i)_{i \in B} \in \mathbb{F}_q^{|B|}$, i.e., the vector obtained by restricting $\mathbf{x}$ to those coordinates i with $i \in B$. The *support* $\text{supp}(\mathbf{x})$ of $\mathbf{x} \in \mathbb{F}_q^{n+1}$ is the set of indices $i \in \mathcal{I}(C)$ with $x_i \neq 0$. An element $\mathbf{c} \in C$ is minimal if there is no $\mathbf{c}' \in C \setminus \{\mathbf{0}\}$ with $\text{supp}(\mathbf{c}')$ a proper subset of $\text{supp}(\mathbf{c})$.

A generator matrix G for an $[n+1,k]$-code C is a matrix with entries in $\mathbb{F}_q$ and that has k columns and $n+1$ rows such that the columns of G jointly constitute an $\mathbb{F}_q$-basis of C. The Hamming-weight $w_H(\mathbf{x})$ of $\mathbf{x} = (x_0, x_1, \ldots, x_n) \in \mathbb{F}_q^{n+1}$ is the number of indices with $x_i \neq 0$. Let d be a positive integer with $0 \leq d \leq n+1$. An $[n+1,k,d]_q$-code C is an $[n+1,k]_q$-code whose minimum distance $d_{\text{min}}(C)$ in the Hamming-metric is *at least* d.[1] If C is an $[n+1,k]_q$-code, then $d_{\text{min}}(C) \leq (n+1) - k + 1 = n - k + 2$ by the Singleton-bound.

The dual $C^\perp$ of C is the "orthogonal complement" of C in $\mathbb{F}_q^{n+1}$ according to the standard scalar product $\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + x_1 y_1 + \cdots + x_n y_n$, where $\mathbf{x} = (x_0, x_1, \ldots, x_n) \in \mathbb{F}_q^{n+1}$ and $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in \mathbb{F}_q^{n+1}$. Thus, $C^\perp$ consists of all $\mathbf{c}^* \in \mathbb{F}_q^{n+1}$ such that $\langle \mathbf{c}, \mathbf{c}^* \rangle = 0$ for all $\mathbf{c} \in C$. If C is an $[n+1,k]_q$-code, then $C^\perp$ is an $[n+1, n+1-k]_q$-code. Note that $d_{\text{min}}(C) + d_{\text{min}}(C^\perp) \leq n+3$.

2.2 Secret Sharing

In this section we give precise definitions of (linear) secret sharing (with strong multiplication). A secret sharing scheme (SSS) $\Sigma = (S_0, S_1, \ldots, S_n)$ is a vector of $n+1$ random variables, where n is a positive integer and where the random variables are all defined on the same finite probability distribution. It is required

[1] The minimum distance of the $[n+1,0]_q$-code $C = \{0\}$ is, by definition, equal to $n+1$.

that $H(S_0|S_1 \dots S_n) = 0$ and that $H(S_0) > 0$. Here $H(\cdot)$ denotes the Shannon-entropy function and $H(\cdot|\cdot)$ denotes conditional entropy. A value taken by S_0 is a "secret", and a value taken by S_i, is a "share" or "the i-th share", $i = 1 \dots n$. Write $\mathcal{P} = \mathcal{P}(\Sigma) = \{1, \dots, n\}$ and $n(\Sigma) = n$. An element $i \in \mathcal{P}$ may sometimes be called "player." If $A \subset \mathcal{P}$ is non-empty, S_A denotes the vector of random variables $(S_i)_{i \in A}$. Note that this bare definition only says that there is some non-constant "secret" that is uniquely determined by the n shares.

The *adversary structure* $\mathcal{A}(\Sigma)$ consists of the empty set as well as any non-empty sets $A \subset \mathcal{P}$ such that $H(S_0|S_A) = H(S_0)$ ("no information about the secret"). The access structure $\Gamma(\Sigma)$ consists of all $B \subset \mathcal{P}$ such that $H(S_0|S_B) = 0$ ("full information about the secret"). By definition, $\mathcal{P} \in \Gamma(\Sigma)$. From a basic information theoretic inequality, $H(S_0) \geq H(S_0|S_B)$ for all non-empty sets $B \subset \mathcal{P}$. Therefore, $\Gamma(\Sigma) \cap \mathcal{A}(\Sigma) = \emptyset$. Let t, r be positive integers. We say that Σ *achieves t-privacy* if $\mathcal{A}(\Sigma)$ includes all sets $A \subset \mathcal{P}$ with $|A| = t$ and we say that Σ *achieves r-reconstruction* if $\Gamma(\Sigma)$ includes all sets $B \subset \mathcal{P}$ with $|B| = r$. Furthermore, $r(\Sigma)$ denotes the minimum integer r for which Σ has r-reconstruction and $t(\Sigma)$ is the largest integer t such that $\mathcal{A}(\Sigma)$ includes all sets $A \subset \mathcal{P}$ with $|A| = t$. A *threshold SSS* is one that achieves t-privacy and $t+1$-reconstruction for some positive integer t. An $(n, t+1, t)$-threshold SSS is one that achieves t-privacy and $t+1$-reconstruction for some integer t, with n being the number of players.

An SSS is *perfect* if $\Gamma(\Sigma) \cup \mathcal{A}(\Sigma) = 2^{\mathcal{P}}$. An element $i \in \mathcal{P}$ is "not a dummy" if there exists a set $B \in \Gamma(\Sigma)$ with $i \in B$ that is minimal with respect to the partial ordering $\Gamma(\Sigma)$ defined by the inclusion-relation. In a perfect SSS it holds that $H(S_i) \geq H(S_0)$ for each $i \in \mathcal{P}$ which is not a dummy ("length of a share is at least length of the secret"). A perfect SSS is ideal if for each such $i \in \mathcal{P}$ equality holds. If $\Gamma(\Sigma)$ does not contain any dummies, it is called *connected.*

A *linear secret sharing scheme* (LSSS) is a tuple $\Sigma = (\mathbb{F}_q, n, e, \mathbf{v}_0, V_1, \dots, V_n)$ where $\mathbb{F}_q$ is a finite field, e, n are positive integers, $\mathbf{v}_0 \in \mathbb{F}_q^e \setminus \{\mathbf{0}\}$, and $V_1, \dots, V_n$ are subspaces of the $\mathbb{F}_q$-vector space $\mathbb{F}_q^e$ such that $\mathbf{v}_0 \in \sum_{i \in \mathcal{P}} V_i$, the subspace of $\mathbb{F}_q^e$ spanned by the V_i's. An LSSS is an SSS in the sense of the definition above if the following conventions are made. Write d_i for the $\mathbb{F}_q$-dimension of V_i, $i = 1 \dots n$. First, for each V_i an $\mathbb{F}_q$-basis $B_i = \{\mathbf{b}_{i1}, \dots, \mathbf{b}_{id_i}\}$ is fixed. Second, the random variables $S_0, S_1, \dots, S_n$ are defined as follows. The secret $s \in \mathbb{F}_q$ is chosen uniformly at random (thereby defining S_0) and $\phi \in \mathrm{Hom}_{\mathbb{F}_q}(\mathbb{F}_q^e, \mathbb{F}_q)$, the $\mathbb{F}_q$-linear map from $\mathbb{F}_q^e$ to $\mathbb{F}_q$, is chosen uniformly random conditioned on $\phi(\mathbf{v}_0) = s$. If $d_i > 0$, the i-th share is $(\phi(\mathbf{b}_{i1}), \dots, \phi(\mathbf{b}_{id_i})) \in \mathbb{F}_q^{d_i}$, thereby defining S_i, $i = 1 \dots n$. For a non-empty set $A \subset \mathcal{P}$, we define $V_A = \sum_{i \in A} V_i$ and we call its $\mathbb{F}_q$-dimension d_A. It can be shown that a non-empty set $B \subset \mathcal{P}$ satisfies $B \in \Gamma(\Sigma)$ if and only if $\mathbf{v}_0 \in V_B$. Equivalently, it can be shown $A \in \mathcal{A}(\Sigma)$ if and only if there exists $\phi \in \mathrm{Hom}_{\mathbb{F}_q}(\mathbb{F}_q^e, \mathbb{F}_q)$ such that $\phi(\mathbf{v}_0) = 1$ and ϕ vanishes on V_A, i.e., $\phi(\mathbf{v}) = 0$ for all $\mathbf{v} \in V_A$. In particular, this means that LSSS are perfect. We define $\dim \Sigma = \sum_{i=1}^{n} d_i$, the dimension of the LSSS. Shamir's scheme is a threshold LSSS.

Given an LSSS Σ we define the *dual access structure* to $\Gamma(\Sigma)$ as $\Gamma(\Sigma)^* = \{A \subset \mathcal{P}, \text{ s.t. } \mathcal{P} \setminus A \notin \Gamma(\Sigma)\}$. For every LSSS Σ over $\mathbb{F}_q$ there exists an LSSS Σ^* over $\mathbb{F}_q$ such that $\dim \Sigma = \dim \Sigma^*$ and $\Gamma(\Sigma^*) = \Gamma(\Sigma)^*$ (see [13, 21]). Note that $r(\Sigma^*) = n - t(\Sigma)$, $t(\Sigma^*) = n - r(\Sigma)$.

Let $\mathbf{v} = (v_1, \ldots, v_e) \in \mathbb{F}_q^e$ and $\mathbf{w} = (w_1, \ldots, w_e) \in \mathbb{F}_q^e$. Then $\mathbf{v} \otimes \mathbf{w} = (v_1 \cdot \mathbf{w}, \ldots, v_e \cdot \mathbf{w}) \in \mathbb{F}_q^{e^2}$ denotes the *Kronecker-product (or tensor-product)* of $\mathbf{v}$ and $\mathbf{w}$. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n, \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$, their *Schur-product* $\mathbf{x} * \mathbf{y} \in \mathbb{F}_q^n$ is defined as $(x_1 y_1, \ldots, x_n y_n)$.

If V and W are subspaces of the $\mathbb{F}_q$-vector space $\mathbb{F}_q^e$, then $V \otimes W$ denotes the subspace of the $\mathbb{F}_q$-vector space $\mathbb{F}_q^{e^2}$ generated by the elements $\mathbf{v} \otimes \mathbf{w}$ with $\mathbf{v} \in V$ and $\mathbf{w} \in W$. If $A \subset \{1, \ldots, n\}$ is non-empty, $\widehat{V}_A$ denotes $\sum_{i \in A} V_i \otimes V_i$, the subspace of the $\mathbb{F}_q$-vector space $\mathbb{F}_q^{e^2}$ spanned by the subspaces $V_i \otimes V_i$. Σ has *t-strong multiplication* ([8, 12, 13]) if the following holds: $0 \leq t \leq n$, Σ achieves t-privacy, $\mathbf{v}_0 \otimes \mathbf{v}_0 \in \widehat{V}_{\mathcal{P}}$, and for each set $B \subset \mathcal{P}$ with $|B| = n - t$, $\mathbf{v}_0 \otimes \mathbf{v}_0 \in \widehat{V}_B$. Σ has *multiplication* if it achieves t-privacy for some $t \geq 1$ and if $\mathbf{v}_0 \otimes \mathbf{v}_0 \in \widehat{V}_{\mathcal{P}}$.

2.3 Algebraic Function Fields and Codes

In this paper we make at some point use of some basic as well as some more advanced results for algebraic function fields. We use the terminology of [27]. For a quick introduction to some of the notions that are needed (function fields, poles, zeroes, divisors, degrees, etc.), please refer to [8]. Let $\mathbb{F}_q$ be a finite field. When we say that $\mathbb{F}$ is an algebraic function field over $\mathbb{F}_q$ we mean that $\mathbb{F}$ is an algebraic function field over $\mathbb{F}_q$ *in one variable* and that $\mathbb{F}_q$ is the *full constant field* of $\mathbb{F}$. $\mathbb{P}_q(\mathbb{F})$ denotes the set of places of degree 1 and $g(\mathbb{F})$ denotes the genus of $\mathbb{F}$. If G is a divisor on $\mathbb{F}$, then $\deg(G)$ denotes its degree and $\mathcal{L}(G) \subset \mathbb{F}$ denotes the Riemann-Roch space of functions $f \in \mathbb{F}$ such that $\operatorname{div}(f) + G$ is an effective divisor or $f = 0$. This is an $\mathbb{F}_q$-vector space.

By the Riemann-Roch Theorem, $\dim_{\mathbb{F}_q} \mathcal{L}(G) = \deg(G) + 1 - g(\mathbb{F}) + \dim_{\mathbb{F}_q} \mathcal{L}(W - G)$, where $\deg(G)$ denotes the degree of the divisor G and where W is any canonical divisor. This implies Riemann's Theorem that $\dim_{\mathbb{F}_q} \mathcal{L}(G) = \deg(G) + 1 - g(\mathbb{F})$ if $\deg(G) > 2g(\mathbb{F}) - 2$. Suppose $\mathbb{P}_q(\mathbb{F}) \geq n + 1$ for some positive integer n. Let $P_0, P_1, \ldots, P_n$ be distinct elements of $\mathbb{P}_q(\mathbb{F})$ and define the divisor $D = P_0 + P_1 + \ldots + P_n$. Suppose G is a divisor on $\mathbb{F}$ such that $\deg(G) > 2 \cdot g(\mathbb{F}) - 2$ and such that the $\operatorname{supp}(G)$, the support of G is disjoint of that of D. Then the $[n+1, k, d]_q$-code $C(G, D)$ (algebraic-geometric Goppa-code or AG-code) is defined [16] as $C(G, D) = \{(f(P_0), f(P_1), \ldots, f(P_n)) \mid f \in \mathcal{L}(G)\}$. By Riemann's Theorem, $k = \deg(G) + 1 - g(\mathbb{F})$, since for any divisor G' it holds that $\mathcal{L}(G') = \{0\}$ if $\deg(G') < 0$, it follows that $d \geq n + 1 - \deg(G)$. The distance of its dual code can be estimated using the Residue Theorem.

Define $N_q(g)$ as the maximum of $|\mathbb{P}_q(\mathbb{F})|$ where $\mathbb{F}$ ranges over all the function fields whose full field of constants is $\mathbb{F}_q$ and whose genus is g. The *Drinfeld-Vladuts upper bound* (see e.g. [27] or [29]) states that for all finite fields $\mathbb{F}_q$, *Ihara's constant* $A(q) \equiv \limsup_{g \to \infty} \frac{N_q(g)}{g(\mathbb{F})}$, satisfies $A(q) \leq \sqrt{q} - 1$. Note that the *Hasse-Weil bound* (see e.g. [27]) states that $||\mathbb{P}_q(\mathbb{F})| - (q + 1)| \leq 2 \cdot g(\mathbb{F}) \sqrt{q}$.

3 Connecting Secret Sharing and Codes

We describe one connection between secret sharing and codes that is particularly relevant to this work. Let C be a linear code over $\mathbb{F}_q$ with $n(C) \geq 1$. Let $i \in \mathcal{I}(C)$. Under further conditions on C to be formulated precisely later on, consider the following perfect secret sharing scheme, denoted $\Sigma(C, i)$, on the player set $\mathcal{P} = \mathcal{I}(C) \setminus \{i\}$. Let $s \in \mathbb{F}_q$ be the secret, and choose $\mathbf{c} = (c_0, c_1, \ldots, c_n) \in C$ uniformly at random such that $c_i = s$. For all $j \in \mathcal{I}(C) \setminus \{i\}$ the share for the j-th player is c_j. In [7] this approach to secret sharing [23, 24] is exploited to achieve LSSS with multiplication (*no* strong multiplication), t-privacy and r-reconstruction and with very good asymptotic properties over fixed finite fields. While in [7] privacy and reconstruction parameters of these LSSS are bounded exclusively in terms of the minimum distance of the codes involved, in the present paper we need a more accurate understanding of these parameters. This is what we will develop first.

Definition 1. *Let n be an integer with $n \geq 1$ and let $\mathbb{F}_q$ be a finite field. For a non-empty set $B \subset \{0, 1, \ldots, n\}$, the $\mathbb{F}_q$-linear projection map π_B is defined as $\pi_B^{q,n+1} : \mathbb{F}_q^{n+1} \longrightarrow \mathbb{F}_q^{|B|}$, $(x_0, x_1, \ldots, x_n) \mapsto (x_i)_{i \in B}$. When q and n are clear from the context, we write π_B instead. Also, if $B = \{i\}$ for some index i, we write π_i instead of $\pi_{\{i\}}$.*

Lemma 1. *Let C be a linear code over $\mathbb{F}_q$ with $n(C) \geq 1$. Let $i \in \mathcal{I}(C)$ and let $B \subset \mathcal{I}(C) \setminus \{i\}$ be a non-empty set. Then there exists a function $\rho_{B,i} : \pi_B(C) \longrightarrow \mathbb{F}_q$ such that $\rho_{B,i}(\pi_B(\mathbf{c})) = \pi_i(\mathbf{c})$ for all $\mathbf{c} \in C$ if and only if $\pi_B(\mathbf{c}) \neq \mathbf{0}$ for all $\mathbf{c} \in C$ with $\pi_i(\mathbf{c}) \neq 0$. If such function $\rho_{B,i}$ exists, it is an $\mathbb{F}_q$-linear map.*

Proof. In the forward direction, suppose $\rho_{B,i}$ exists. Then it is an $\mathbb{F}_q$-linear map, since $\rho_{B,i}(\lambda\mathbf{c} + \mu\mathbf{c}') = \lambda \cdot \rho_{B,i}(\mathbf{c}) + \mu \cdot \rho_{B,i}(\mathbf{c}')$ for all $\lambda, \mu \in \mathbb{F}_q$, $\mathbf{c}, \mathbf{c}' \in C$, by linearity of C. Suppose there is $\mathbf{c} \in C$ with $\pi_i(\mathbf{c}) \neq 0$ and $\pi_B(\mathbf{c}) = \mathbf{0}$. Then, by $\mathbb{F}_q$-linearity of the map, $\rho_{B,i}(\pi_B(\mathbf{c})) = \rho_{B,i}(\mathbf{0}) = 0 \neq \pi_i(\mathbf{c})$, a contradiction. In the other direction, suppose $\rho_{B,i}$ does not exist. Then there exist $\mathbf{c}, \mathbf{c}' \in C$ such that $\pi_i(\mathbf{c}) \neq \pi_i(\mathbf{c}')$ yet $\pi_B(\mathbf{c}) = \pi_B(\mathbf{c}')$. Then $\pi_i(\mathbf{c} - \mathbf{c}') \neq 0$ and $\pi_B(\mathbf{c} - \mathbf{c}') = \mathbf{0}$ by linearity of C. △

Note that by linearity of C, the lemma above also holds when $\pi_i(\mathbf{c}) \neq 0$ is replaced by $\pi_i(\mathbf{c}) = 1$.

Definition 2. *Notation being as in Lemma 1, we say that (i, B) is a* reconstruction-pair *if $\rho_{B,i}$ exists.*

Corollary 1. *Let C be a linear code over $\mathbb{F}_q$ with $n(C) \geq 1$ and let $i \in \mathcal{I}(C)$. Let $\mathbf{u}_i \in \mathbb{F}_q^{n(C)+1}$ denote the i-th unit vector, i.e., $(\mathbf{u}_i)_j = 1$ if $i = j$ and $(\mathbf{u}_i)_j = 0$ if $i \neq j$. Then:*

- *$(i, \mathcal{I}(C) \setminus \{i\})$ is a reconstruction-pair if and only if $\mathbf{u}_i \notin C$.*
- *$\pi_i(C) \neq \{0\}$ if and only if $\mathbf{u}_i \notin C^{\perp}$.*

– *For all $j \in \mathcal{I}(C)$, $(j, \mathcal{I}(C) \setminus \{j\})$ is a reconstruction-pair and $\pi_j(C) \neq \{0\}$ if and only if $d_{min}(C) > 1$ and $d_{min}(C^\perp) > 1$.*

Definition 3. *Let C be a code over $\mathbb{F}_q$ with $n(C) \geq 1$. Let $i \in \mathcal{I}(C)$ and suppose $\mathbf{u}_i \notin C$. If $\mathbf{u}_i \in C^\perp$, then define $r_i(C) = 0$. Else, define $r_i(C)$ as the smallest positive integer ρ_i such that for all $B \subset \mathcal{I}(C) \setminus \{i\}$ with $|B| = \rho_i$ it holds that (i, B) is a reconstruction-pair.*

Note that by Corollary 1, the value $r_i(C)$ is well-defined, and satisfies $0 \leq r_i(C) \leq n(C)$.

Definition 4. *Let C be a code over $\mathbb{F}_q$ with $n(C) \geq 1$. Let $i \in \mathcal{I}(C)$ and suppose $\mathbf{u}_i \notin C^\perp$. Define $t_i(C)$ as the largest positive integer τ_i such that for each set $A \subset \mathcal{I}(C) \setminus \{i\}$ with $|A| = \tau_i$ it holds that (i, A) is not a reconstruction-pair. Equivalently, this is satisfied if and only if there exists $\mathbf{c} \in C$ with $\pi_i(\mathbf{c}) = 1$ and $\pi_A(\mathbf{c}) = \mathbf{0}$. If no such integer exists, $t_i(C) = 0$ by definition.*

Note that by Corollary 1, the value $t_i(C)$ is well-defined, and satisfies $0 \leq t_i(C) < n(C)$.

Lemma 2. *Let C be a linear code over $\mathbb{F}_q$ such that $\{\mathbf{0}\} \subsetneq C, C^\perp \subsetneq \mathbb{F}_q^{n(C)+1}$. Then $d_{min}(C^\perp) = m + 1$, where m is the largest positive integer such that for all non-empty sets $B \subset \{0, 1, \ldots, n\}$ with $|B| = m$, it holds that $\pi_B(C) = \mathbb{F}_q^{|B|}$.*

Proof. The conditions imply that $n(C) \geq 1$. Write $d^\perp = d_{\text{min}}(C^\perp)$. For a non-empty set $B \subset \mathcal{I}(C)$, write $W_B = \pi_B(C) \subset \mathbb{F}_q^{|B|}$. Clearly, $W_B \neq \mathbb{F}_q^{|B|}$ if and only if $W_B^\perp \neq \{\mathbf{0}\}$. This latter condition equivalent to the existence of some $\mathbf{c}^* \in C^\perp \setminus \{\mathbf{0}\}$ with $\text{supp}(\mathbf{c}^*) \subset B$, for which we have that $w_H(\mathbf{c}^*) \leq |B|$. Thus, for all $B \subset \mathcal{I}(C)$ with $|B| \leq d^\perp - 1$, it must hold that $W_B = \mathbb{F}_q^{n+1}$. On the other hand, since $C^\perp \neq \{\mathbf{0}\}$, an element $\mathbf{c}^* \in C^\perp \setminus \{\mathbf{0}\}$ can be selected with minimal weight $d^\perp$. Define $B = \text{supp}(\mathbf{c}^*)$. Then $|B| = d^\perp$, and by the characterization above, $W_B \neq \mathbb{F}_q^{n+1}$. △

Lemma 3. *Let C be a code over $\mathbb{F}_q$ with $n(C) \geq 1$ and let $i \in \mathcal{I}(C)$. Then:*

1. *If $\mathbf{u}_i \notin C$, then $r_i(C) \leq n(C) - d_{min}(C) + 2$.*
2. *If $d_{min}(C) > 1$, then $\max_{j \in \mathcal{I}(C)} r_j(C) = n(C) - d_{min}(C) + 2$.*
3. *If $\mathbf{u}_i \notin C^\perp$, then $d_{min}(C^\perp) - 2 \leq t_i(C)$.*
4. *If $d_{min}(C^\perp) > 1$ then $d_{min}(C^\perp) - 2 = \min_{i \in \mathcal{I}(C)} t_i(C)$.*

Proof. As to Claim 1,if $d_{\text{min}}(C) \leq 2$, there is nothing to prove. Else, if we "prune" the code C at $d_{\text{min}}(C) - 2$ coordinates (not including i), then we get a code C' with $d_{\text{min}}(C') > 1$. The claim now follows from Corollary 1. As to Claim 2, $d_{\text{min}}(C) > 1$, then r_i is well-defined for all $i \in \mathcal{I}(C)$. Select an element $\mathbf{c}$ in C of minimal weight $d_{\text{min}}(C)$. Take any $i \in \text{supp}(\mathbf{c})$ and define $B = \mathcal{I}(C) \setminus \text{supp}(\mathbf{c})$. Clearly $|B| = n(C) - d_{\text{min}}(C) + 1$ and (i, B) is not a reconstruction-pair because $\pi_i(\mathbf{c}) \neq 0$ and $\pi_B(\mathbf{c}) = 0$. Therefore $r_i(C) \geq n(C) - d_{\text{min}}(C) + 2$, which was what remained to be proved. As to Claim 3, this follows directly from Lemma 2. As to Claim 4, if $d_{\text{min}}(C^\perp) > 1$, then t_i is well-defined for all $i \in \mathcal{I}(C)$. So if $B \subset \mathcal{I}(C)$ is any set with $|B| = 1 + \min_{i\in\mathcal{I}(C)} t_i(C)$, then for each $j \in B$ there exists $\mathbf{c} \in C$ such that $\pi_j(\mathbf{c}) = 1$ and $\pi_{B'}(\mathbf{c}) = \mathbf{0}$, where $B' = B\backslash\{j\}$. Thus, $\pi_B(C) = \mathbb{F}_q^{|B|}$, and by Lemma 2, $|B| = 1 + \min_{i\in\mathcal{I}(C)} t_i(C) \leq d_{\text{min}}(C^\perp) - 1$. Hence, $d_{\text{min}}(C^\perp) - 2 \geq \min_{i\in\mathcal{I}(C)} t_i(C)$, which was what remained to be proved. $\triangle$

Definition 5. *Let C be a linear code over $\mathbb{F}_q$ with $n(C) \geq 1$. We define $I(C)$ as the set consisting of all indices $i \in \mathcal{I}(C)$ such that $\mathbf{u}_i \notin C$ and $\mathbf{u}_i \notin C^\perp$. $\mathcal{C}(\mathbb{F}_q)$ is the collection of all linear codes C over $\mathbb{F}_q$ such that $I(C) \neq \emptyset$.*

Note that $I(C) \neq \emptyset$ implies $n(C) \geq 1$ and that $I(C) = \mathcal{I}(C)$ if and only if $d_{\text{min}}(C) > 1$ and $d_{\text{min}}(C^\perp) > 1$. For completeness we state the following straightforward characterization of $\mathcal{C}(\mathbb{F}_q)$. If C is a linear code over $\mathbb{F}_q$ with $n(C) \geq 1$ and if $I(C) = \emptyset$, then it holds for all $i \in \mathcal{I}(C)$ that the i-th coordinate of elements of C is always equal to zero or that the i-th unit vector $\mathbf{u}_i \in C$. After permutating indices, if necessary, the set C is then equal to a Cartesian product $\mathbb{F}_q \times \cdots \times \mathbb{F}_q \times \{\mathbf{0}\} \times \cdots \times \{\mathbf{0}\}$. On the other hand, if C decomposes as above, then clearly $I(C) = \emptyset$.

Theorem 1. *Let $C \in \mathcal{C}(\mathbb{F}_q)$ and let $i \in I(C)$. Suppose $\mathbf{c} \in C$ is chosen uniformly at random. Then the following holds.*

1. *$\pi_i(C) \in \mathbb{F}_q$ has the uniform distribution.*
2. *("r-reconstruction") If $B \subset \mathcal{I}(C) \setminus \{i\}$ with $|B| \geq r_i(C)$, then $\pi_B(\mathbf{c})$ determines $\pi_i(\mathbf{c})$ uniquely with probability 1. Thus, $\Sigma(C, i)$ has $r_i(C)$-reconstruction.*
3. *If $d_{min}(C) > t + 1$ for some positive integer t, then $\Sigma(C, i)$ has $(n - t)$-reconstruction.*
4. *("t-privacy") Suppose $t_i(C) \geq 1$. If $A \subset \mathcal{I}(C) \setminus \{i\}$ is non-empty and $|A| \leq t_i(C)$, then $\pi_i(\mathbf{c})$ has the uniform distribution on $\mathbb{F}_q$ and $\pi_A(\mathbf{c})$ is distributed independently from $\pi_i(\mathbf{c})$. Thus, $\Sigma(C, i)$ has $t_i(C)$-privacy.*
5. *If $d_{min}(C^\perp) > t + 1$ for some positive integer t, then $\Sigma(C, i)$ has t-privacy.*
6. *Suppose $d_{min}(C^\perp) > 1$. The largest positive integer m such that for all $A \subset \mathcal{I}(C)$ with $|A| = m$ it holds that $\pi_A(\mathbf{c}) \in \mathbb{F}_q^{|A|}$ has the uniform distribution, satisfies $m = d_{min}(C^\perp) - 1$.*

Proof. As to Claim 1, $i \in I(C)$ implies in particular that for each $x \in \mathbb{F}_q$ there exists $\mathbf{c} \in C$ such that $\pi_i(\mathbf{c}) = x$. Moreover, their number is equal to the cardinality of the kernel of the map π_i. Hence this number does not depend on x and the claim follows. Claim 2 follows from the definition of $r_i(C)$. As to Claim 3, this follows from Lemma 3 plus pruning. As to Claim 4, if $|A| \leq t_i(C)$, then there exists $\mathbf{c}'' \in C$ with $\pi_i(\mathbf{c}'') = 1$ and $\pi_A(\mathbf{c}'') = \mathbf{0}$. This implies that for each $(x, \mathbf{y})$ with $x \in \mathbb{F}_q$ and $\mathbf{y} \in \pi_B(C)$, there exists $\mathbf{c}'' \in C$ with $\pi_i(\mathbf{c}'') = x$ and $\pi_A(\mathbf{c}'') = \mathbf{y}$. More precisely, their number is equal to the cardinality of the kernel of the map $\pi_{A\cup\{i\}}$, and the claim follows. As to Claim 5, this follows from

Lemma 3. As to Claim 6, by Lemma 2 it holds that for each $\mathbf{y} \in \mathbb{F}_q^{|A|}$, there exists $\mathbf{c} \in C$ with $\pi_A(\mathbf{c}) = \mathbf{y}$. Their number is equal to the cardinality of the kernel of the map π_A, and, as maximality also follows from Lemma 2, the claim follows. $\triangle$

Remark 1. *Let $C \in \mathcal{C}(\mathbb{F}_q)$ and let $i \in I(C)$. Then $\Sigma(C, i)$ can be viewed as an LSSS.*

Proof. Assume for simplicity in notation that $i = 0$. Choose a generator matrix G for the code C, i.e., a matrix with k columns and $n + 1$ rows such that the columns jointly constitute an $\mathbb{F}_q$- basis of C. Write $\mathbf{v}_0$ for the top row, write $\mathbf{v}_i$ for the i-th row below, and write V_i for the $\mathbb{F}_q$-vector space spanned by it, which is one-dimensional as $\mathbf{v}_i \neq \mathbf{0}$ $(i = 1 \ldots n)$. Since $d_{\text{min}}(C) > 1$, it follows by Lemma 1 that there exists a vector $\mathbf{x} = (x_0, x_1, \ldots, x_n)^T \in \mathbb{F}_q^{n+1}$ such that $G^T \mathbf{x} = \mathbf{0}$ and $x_0 = 1$, where G^T denotes the transpose of G. Thus, $\mathbf{v}_0$ is in the $\mathbb{F}_q$-linear span of the $\mathbf{v}_i$, $i = 1, \ldots, n$. The parameter e from the LSSS definition is equal to k, the dimension of C. Thus, $(\mathbb{F}_q, n, e, \mathbf{v}_0, V_1, \ldots, V_n)$ thus defined is an LSSS by definition. The secret sharing scheme it generates (see Section 2) is identical to choosing $\mathbf{b} \in \mathbb{F}_q^{n+1}$ at random and setting $(s_0, s_1, \ldots, s_n)^T = G\mathbf{b}$. Since G is a generator matrix of C, this secret sharing scheme is identical to $\Sigma(C, i)$. $\triangle$

4 Strongly Multiplicative LSSS from Codes

In this section we define a special class of codes that imply strongly multiplicative LSSS.

Definition 6. *For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n, \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$, their* Schur-product *$\mathbf{x} * \mathbf{y} \in \mathbb{F}_q^n$ is defined as $(x_1 y_1, \ldots, x_n y_n)$. Let C be a linear code over $\mathbb{F}_q$. The linear code $\widehat{C}$ over $\mathbb{F}_q$ is the linear code $\mathbb{F}_q < \{\mathbf{x} * \mathbf{y}\}_{\mathbf{x},\mathbf{y} \in C} >$, i.e., the $\mathbb{F}_q$-linear span of the vectors of the form $\mathbf{x} * \mathbf{y}$ with $\mathbf{x}, \mathbf{y} \in C$.*

Lemma 4. *Let C be a linear code over $\mathbb{F}_q$ with $n(C) \geq 1$ and let $i \in \mathcal{I}(C)$. Then: 1) $n(\widehat{C}) = n(C)$. 2) $\mathbf{u}_i \notin \widehat{C}$ implies $\mathbf{u}_i \notin C$. 3) $\mathbf{u}_i \notin C^\perp$ if and only if $\mathbf{u}_i \notin (\widehat{C})^\perp$. 4) $I(\widehat{C}) \subset I(C)$. 5) $1 \leq d_{min}(\widehat{C}) \leq d_{min}(C)$.*

Generally, $\mathbf{u}_i \notin C$ does not necessarily imply $\mathbf{u}_i \notin \widehat{C}$.

Definition 7. *$\mathcal{C}^\dagger(\mathbb{F}_q)$ denotes the set of all $\mathbb{F}_q$-linear codes C with $I(\widehat{C}) \neq \emptyset$.*

Note that $\mathcal{C}^\dagger(\mathbb{F}_q) \neq \emptyset$ for all finite fields $\mathbb{F}_q$.

Definition 8. *For $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$, $\widehat{t}(C) = \max_{i \in I(\widehat{C})} \min\{t_i(C), n(\widehat{C}) - r_i(\widehat{C})\}$. The LSSS $\Sigma(C)$ is by definition $\Sigma(C, i)$ where i is the smallest index where this maximum is attained. Write i_s for this index.*

Note that $r_i(\widehat{C})$ is well-defined in the definition of $\widehat{t}(C)$ since $\widehat{C} \in \mathcal{C}(\mathbb{F}_q)$ and $i \in I(\widehat{C})$.

Generally, $C \in \mathcal{C}(\mathbb{F}_q)$ does not even need to imply $C \in \mathcal{C}^{\dagger}(\mathbb{F}_q)$. In fact, only for special classes of codes one seems to be able to bound $\widehat{t}(C)$ non-trivially, sometimes in combination with this Corollary to Theorem 1.

Corollary 2. *Let $C \in C^{\dagger}(\mathbb{F}_q)$. Suppose that $d_{min}(C^{\perp}) > t+1$ and $d_{min}(\widehat{C}) > t+1$ for some integer $t \geq 1$. Then $\widehat{t}(C) \geq t$.*

Lemma 5. *Let e be a positive integer. Then:*

- $\langle \mathbf{v} \otimes \mathbf{w}, \mathbf{a} \otimes \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{a} \rangle \cdot \langle \mathbf{w}, \mathbf{b} \rangle$, *for all* $\mathbf{v}, \mathbf{w}, \mathbf{a}, \mathbf{b} \in \mathbb{F}_q^e$.
- *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^{e^2}$. *Then* $\mathbf{x} = \mathbf{y}$ *if and only if* $\langle \mathbf{x}, \mathbf{a} \otimes \mathbf{b} \rangle = \langle \mathbf{y}, \mathbf{a} \otimes \mathbf{b} \rangle$ *for all* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^e$.

Proof. The definitions of tensor product and scalar product imply the first claim. The second follows by combination of the bilinearity of the scalar product and the facts that the $\mathbb{F}_q$-linear span of the vectors $\mathbf{a} \otimes \mathbf{b}$ with $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^e$ is equal to $\mathbb{F}_q^{e^2}$ and that the scalar-product with a given vector is always zero if and only if that vector equals the zero-vector. $\triangle$

Theorem 2. *Let $C \in \mathcal{C}^{\dagger}(\mathbb{F}_q)$. Suppose $\widehat{t}(C) \geq 1$ and let t be an integer with $1 \leq t \leq \widehat{t}(C)$. Then: $t \leq \frac{1}{3} \cdot (n(C) - 1)$, $\Sigma(C)$ has $(n(C) - 2t)$-reconstruction, and $\Sigma(C)$ has t-strong multiplication.*

Proof. We first argue t-strong multiplication. Assume w.l.o.g. that $i_s = 0$ (Definition 8). By Theorem 1, $\Sigma(C)$ satisfies t-privacy. Write $n = n(\widehat{C})$ $(= n(C))$. Since $\widehat{t}(C) \geq 1$, $r_0(\widehat{C}) < n$. Now choose a generator matrix G for C. Write $\mathbf{v}_0$ for its top row, write $\mathbf{v}_i$ for the i-th row below and write V_i for the one-dimensional space V_i spanned by it, $i = 1 \ldots n$. Let $B \subset \{1, \ldots, n\}$ be a nonempty set. First, note that there exists a vector $\lambda \in \mathbb{F}_q^n$ such that $\sum_{i \in B} \pi_i(\lambda)(\mathbf{v}_i \otimes \mathbf{v}_i) = \mathbf{v}_0 \otimes \mathbf{v}_0$ if and only if $\langle \sum_{i \in B} \pi_i(\lambda)(\mathbf{v}_i \otimes \mathbf{v}_i), \mathbf{b} \otimes \mathbf{b}' \rangle = \langle \mathbf{v}_0 \otimes \mathbf{v}_0, \mathbf{b} \otimes \mathbf{b}' \rangle$ for all $\mathbf{b}, \mathbf{b}' \in \mathbb{F}_q^e$. Indeed, the forward direction follows by rewriting and the reverse direction follows from Lemma 5 (2nd item). By re-writing and by using Lemma 5 (1st item), this is equivalent to $\sum_{i \in B} \pi_i(\lambda) \langle \mathbf{v}_i, \mathbf{b} \rangle \langle \mathbf{v}_i, \mathbf{b}' \rangle = \langle \mathbf{v}_0, \mathbf{b} \rangle \langle \mathbf{v}_0, \mathbf{b}' \rangle$ for all $\mathbf{b}, \mathbf{b}' \in \mathbb{F}_q^e$. This may be rewritten as $\langle (G\mathbf{b}) * (G\mathbf{b}'), \mathbf{y} \rangle = 0$ for all $\mathbf{b}, \mathbf{b}' \in \mathbb{F}_q^e$, where $\pi_0(\mathbf{y}) = -1$, $\pi_i(\mathbf{y}) = \pi_i(\lambda)$ if $i \in B$, and $\pi_i(\mathbf{y}) = 0$ for all other indices. Equivalently, $\sum_{i \in B} \lambda_i \pi_i(\mathbf{c}) \pi_i(\mathbf{c}') = \pi_0(\mathbf{c}) \pi_0(\mathbf{c}')$ for all $\mathbf{c}, \mathbf{c}' \in C$, since G is a generator matrix of C. By definition of t, there exists, for each choice of B with $|B| = n - t$, a vector $\lambda \in \mathbb{F}_q^n$ such that the latter condition is satisfied for the set B. We conclude that t-strong multiplication holds, as desired. As to the remaining claims, let $B \subset \{1, \ldots, n\}$ be such that $|B| = n - 2t$. Write $A = \{1, \ldots, n\} \setminus B$. Choose a disjoint partition $A_0 \cup A_1 = A$ with $|A_0| = |A_1| = t$. By Lemma 2 there exists $\mathbf{c}' \in C$ such that $\pi_0(\mathbf{c}') = 1$ and $\pi_{A_0}(\mathbf{c}') = \mathbf{0}$. Let $\mathbf{c} \in C$ be arbitrary and consider the vector $\mathbf{c} * \mathbf{c}' \in \widehat{C}$. Note that this vector has coordinates equal to zero at those indices i with $i \in A_0$. Since $\widehat{C}$ has $(n - t)$-reconstruction, there exists a vector $\mathbf{x} \in \mathbb{F}_q^n$ such that it has coordinates equal to zero at those indices i with $i \in A_1$ and $\pi_0(\mathbf{c} * \mathbf{c}') = \sum_{i=1}^n \pi_i(\mathbf{x})\pi_i(\mathbf{c} * \mathbf{c}')$. It now follows that $\pi_0(\mathbf{c}) = \sum_{i \in B}(\pi_i(\mathbf{x}) \cdot \pi_i(\mathbf{c}'))\pi_i(\mathbf{c})$, for all $\mathbf{c} \in C$. Thus, there is $(n-2t)$-reconstruction. Since there is also t-privacy, it follows that $t \leq \frac{1}{3}(n(C)-1)$. Finally, the remark about fulfilment of the conditions follows from Theorem 1. $\triangle$

We introduce the notion of asymptotic optimal corruption tolerance for the class of codes $\mathcal{C}^\dagger(\mathbb{F}_q)$.

Definition 9. *Let $\mathbb{F}_q$ be a finite field. For $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$, we define $\widehat{\tau}(C) = \frac{3 \cdot \widehat{t}(C)}{n(C)-1}$. We call value $\widehat{\tau}(C)$ the* corruption tolerance *of the code C.*

Note that $0 \leq \widehat{\tau}(C) \leq 1$ always, where the upper bound follows from Theorem 2. Let t, n be positive integers with $3t < n$ and let $\mathbb{F}_q$ be a finite field with $q > n$. If C is a polynomial evaluation code ("Reed-Solomon code") over $\mathbb{F}_q$ of length $n+1$, defined from evaluation of the polynomials of degree at most t, then $\widehat{\tau}(C) = 1$, and, of course, $\Sigma(C)$ is Shamir's $(n, t+1, t)$-threshold LSSS.

Definition 10. *(*Asymptotic optimal corruption tolerance*). Let $\mathbb{F}_q$ be a finite field. Then we define $\widehat{\tau}(q) = \limsup_{C \in \mathcal{C}^\dagger(\mathbb{F}_q)} \widehat{\tau}(C)$.*

Note that $\widehat{\tau}(C) = 1$ implies that $\Sigma(C)$ is an $(n, t+1, t)$-threshold LSSS over $\mathbb{F}_q$, with $n = 3t+1$. For fixed q there are only finitely many $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$ such that $\widehat{\tau}(C) = 1$ (the proof for this statement is easily extracted from [8]; in fact, in Section 8 we prove a stronger statement). Since for each length there are only a finite number of codes of that length when $\mathbb{F}_q$ is constant, this means that for each $\epsilon > 0$ there exists an infinite family of codes $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$ with $\ell(C)$ tending to infinity and $|\widehat{\tau}(q) - \widehat{\tau}(C)| < \epsilon$.

5 Bounding $\widehat{\tau}(q)$ Away from Zero for Arbitrary $\mathbb{F}_q$

The main result of this section is the fact that $\widehat{\tau}(q) > 0$ for every finite field $\mathbb{F}_q$. First, we need to restate and reprove part of the results (Theorems 3 and 6) of [8] on algebraic geometric strongly multiplicative secret sharing in the technical framework of the present paper. Throughout this section $\mathbb{F}_q$ denotes the finite field with q elements.

Theorem 3. *(Chen and Cramer [8]) Let $\mathbb{F}$ be an algebraic function field over $\mathbb{F}_q$. Suppose $|\mathbb{P}_q(\mathbb{F})| > 4(g(\mathbb{F})+1)$. Let t, n be any integers such that $1 \leq t < n$, $|\mathbb{P}_q(\mathbb{F})| \geq n+1$, and $3t < n - 4 \cdot g(\mathbb{F})$. Then there exists a code $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$ such that $\ell(C) = n+1$ and $\widehat{t}(C) \geq t$. In particular, $\Sigma(C)$ has t-strong multiplication.*

Proof. By the condition on $|\mathbb{P}_q(\mathbb{F})|$, there exist integers t, n satisfying the constraints from the theorem. Now fix such t, n. By Corollary 2, it is sufficient to show the existence of $C \in \mathcal{C}^\dagger(\mathbb{F}_q)$ with $\ell(C) = n+1$, $d_{\min}(C^\perp) > t+1$ and $d_{\min}(\widehat{C}) > t+1$. Write $g = g(\mathbb{F})$. Let $P_0, P_1, \ldots, P_n \in \mathbb{P}_q(\mathbb{F})$ be distinct places of degree 1, and define the divisor $D = \sum_{i=0}^{n} P_i$. Choose a divisor G with $\text{supp}(G) \cap \text{supp}(D) \neq \emptyset$ and $\deg(G) = 2g + t$. This is possible by the Weak Approximation Theorem (see e.g. [27]). Alternatively, in case $|\mathbb{P}_q(\mathbb{F})| > n + 1$, select a place $Q \in \mathbb{P}_q(\mathbb{F}) \setminus \{P_0, P_1, \ldots, P_n\}$ and define $G = (2g + t) \cdot Q$. In any case, it holds that $\dim_{\mathbb{F}_q}(\mathcal{L}(G)) = g + t + 1$. Next, define C as the evaluation code $C(D; G)$. Arbitrarily choose $i \in \{0, 1, \ldots n\}$, $A \subset \{0, 1, \ldots n\} \setminus \{i\}$ with $|A| = t$. Since $2g - 2 < \deg(G - P_i - \sum_{j \in A} P_j) < \deg(G - \sum_{j \in A} P_j)$, it holds that $\dim_{\mathbb{F}_q}(\mathcal{L}(G - P_i - \sum_{j \in A} P_j)) < \dim_{\mathbb{F}_q}(\mathcal{L}(G - \sum_{j \in A} P_j))$. Hence, there exists $f \in \mathcal{L}(G)$ such that $f(P_i) = 1$ and $f(P_j) = 0$ for all $j \in A$. In particular, for C as well as for $\widehat{C}$ it holds that the i-th coordinate is not always zero. Second, since $f \cdot g \in \mathcal{L}(2G)$ if $f, g \in \mathcal{L}(G)$, it follows that $\widehat{C} \subset C(D; 2G)$. From $\deg(2G) = 4g + 2t$ and $4g < n - 3t$, it follows that $d_{\text{min}}(\widehat{C}) \geq n + 1 - \deg(2G) = n + 1 - (4g + 2t) > t + 1$. In particular, it follows that $\mathbf{u}_i \notin \widehat{C}$. We conclude that $C \in \mathcal{C}^{\dagger}(\mathbb{F}_q)$ and $I(C) = \mathcal{I}(C)$, and using Theorem 1, that $t \leq \min_{0 \leq j \leq n} t_j(C) = d_{\text{min}}(C^{\perp}) - 2$. Hence $d_{\text{min}}(C^{\perp}) > t + 1$. By Corollary 2, $\widehat{t}(C) \geq t$. The claim about t-strong multiplicativity of $\Sigma(C)$ follows from Theorem 2. $\triangle$

It follows from this theorem that $\widehat{\tau}(q) > 0$ if $A(q) > 4$, where $A(q)$ is Ihara's constant (see Section 2). Recall that the Drinfeld-Vladuts bounds states that $A(q) \leq \sqrt{q} - 1$. For our purposes, however, we need a lower bound. Ihara [18] has shown that if q is a square, then $A(q) \geq \sqrt{q} - 1$, so that the Drinfeld-Vladuts bound is sharp. Later, Garcia and Stichtenoth [15] showed this result by more explicit methods (see also [1] for recent results over cubic fields).

Theorem 4. *(Ihara [18], Garcia and Stichtenoth [15]) Let $\mathbb{F}_q$ be a finite field and let q be a square. Then $A(q) = \sqrt{q} - 1$. More precisely, there exists an infinite family of algebraic function fields (in one variable) $\{\mathbb{F}^{(m)}\}_{m \geq 1}$ over $\mathbb{F}_q$ such that for all $m \geq 1$, $\mathbb{F}_q$ is the full constant field of $\mathbb{F}^{(m)}$, $|\mathbb{P}_q(\mathbb{F}^{(m)})| \geq (q - \sqrt{q})\sqrt{q}^{m-1}$ and $g(\mathbb{F}^{(m)}) \leq \sqrt{q}^{m}$.*

Theorem 5. *(Serre [25]) There exists a positive constant $c_* \in \mathbb{R}$ such that for all finite fields $\mathbb{F}_q$ we have $A(q) \geq c_* \cdot \log q$.*

Combining Theorems 3, 4 and 5, we can bound $\widehat{\tau}(q)$ away from zero if either q is a square or q is extremely large[2]. Also, we see that $\widehat{\tau}(q)$ tends to 1 if $|\mathbb{F}_q|$ tends to infinity.

Theorem 6. *(Chen and Cramer [8])*

- $\widehat{\tau}(q) \geq 1 - \frac{4}{A(q)}$ *if $\mathbb{F}_q$ is a finite field with $A(q) > 4$. In particular, $A(q) > 4$ if q is large enough, more precisely, if $q > 2^{\frac{4}{c}}$. Here, $c \in \mathbb{R}$ is any positive constant so that Theorem 5 holds with $c_* = c$[3].*
- $\widehat{\tau}(q) \geq 1 - \frac{4}{\sqrt{q}-1}$ *for all finite fields $\mathbb{F}_q$ such that $q \geq 49$ and q is a square.*

[2] The results in [8] only considered the case $q \geq 49$ with q a square. But the combination of Theorem 3 with Theorem 5 is straightforward, so we attribute that in essence to [8].

[3] Currently, $c_* \geq \frac{1}{91}$ is the best known approximation, see [29].

– $\lim_{|\mathbb{F}_q|\to\infty} \widehat{\tau}(q) = 1$, *where* $\mathbb{F}_q$ *ranges over all finite fields.*

Thus, it remains to bound $\widehat{\tau}(q)$ away from zero in the cases where q is small ($2 \leq q < 49$) or $q > 49$ is not a square and q is at most moderately large. We resolve this by means of a dedicated field-descent that allows us to lower bound $\widehat{\tau}(q)$ as a function of $\widehat{\tau}(q^m)$. At its heart it uses the following notion.

Definition 11. *A* multiplication-friendly embedding of the extension field $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ *is a triple* (r, σ, ψ) *where* r *is a positive integer and where* $\sigma : \mathbb{F}_{q^m} \to \mathbb{F}_q^r$ *and* $\psi : \mathbb{F}_q^r \to \mathbb{F}_{q^m}$ *are* $\mathbb{F}_q$*-linear maps such that* $xy = \psi(\sigma(x) * \sigma(y))$ *for all* x, y *in* $\mathbb{F}_{q^m}$*. The integer* r *is called the* expansion.

Note that σ is an *injective* $\mathbb{F}_q$-linear map between $\mathbb{F}_q$-vectorspaces: $\sigma(x) = \sigma(y)$ implies $x = x \cdot 1 = \psi(\sigma(x) * \sigma(1)) = \psi(\sigma(y) * \sigma(1)) = y \cdot 1 = y$. Note that this notion has been studied in the context of asymptotic arithmetic complexity (see [4] and [28]). We can now state and prove our field-descent theorem. Elementary constructions of multiplication-friendly embeddings are given afterwards.

Theorem 7. *Let* t, r *be integers with* $t, r \geq 1$*. Suppose* $C \in \mathcal{C}^\dagger(\mathbb{F}_{q^m})$ *with* $\widehat{t}(C) \geq t$ *and suppose there exists a multiplication-friendly embedding of* $\mathbb{F}_{q^m}$ *over* $\mathbb{F}_q$ *with expansion* r*. Then there exists* $C_1 \in \mathcal{C}^\dagger(\mathbb{F}_q)$ *such that* $n(C_1) = r \cdot n(C)$ *and* $\widehat{t}(C_1) \geq t$*.*

Proof. Write $n = n(C)$. W.l.o.g., $i_s = 0$ (Definition 8), i.e., $\widehat{t}(C)$ is attained for $i = 0$. In particular, $0 \in I(\widehat{C})$. Let π_B denote the projection $\pi_B^{(q^m,n+1)}$ and let π'_B denote the projection $\pi_B^{(q,rn+1)}$ (Definition 1). For an index-set $\mathcal{I}()$, $\mathcal{I}^*()$ denotes $\mathcal{I}() \setminus \{0\}$. Consider the set $G = C \cap (\mathbb{F}_q \bigoplus (\mathbb{F}_{q^m})^n)$, i.e., all $\mathbf{c} \in C$ with $\pi_0(\mathbf{c}) \in \mathbb{F}_q$. Note that $G \neq \emptyset$, G is not an $\mathbb{F}_{q^m}$-linear code, but G is an $\mathbb{F}_q$-linear subspace of the $\mathbb{F}_{q^m}$-linear code C. Let (r, σ, ψ) be a multiplication-friendly embedding of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Define the $\mathbb{F}_q$-linear map $\chi : \mathbb{F}_q \bigoplus (\mathbb{F}_{q^m})^n \to (\mathbb{F}_q)^{1+rn}$ by $(c_0, c_1, \ldots, c_n) \mapsto (c_0, \sigma(c_1), \ldots, \sigma(c_n))$. Now define the $\mathbb{F}_q$-linear code C_1 as $C_1 = \chi(G) \subset \mathbb{F}_q^{rn+1}$. We first show $C_1 \in C^\dagger(\mathbb{F}_q)$. Write $\mathbf{u}_0 = (1, 0, \ldots, 0) \in \mathbb{F}_q^{n+1}$ and $\mathbf{u}'_0 = (1, 0, \ldots, 0) \in \mathbb{F}_q^{rn+1}$. Since $0 \in I(\widehat{C})$, $\mathbf{u}_0 \notin C^\perp$ by Lemma 4, or equivalently, there is $\mathbf{c} \in G$ with $\pi_0(\mathbf{c}) = 1$. Since $\pi'_0(\chi(\mathbf{c})) = 1$, $\mathbf{u}'_0 \notin (C_1)^\perp$, and by Lemma 4, $\mathbf{u}'_0 \notin (\widehat{C}_1)^\perp$. Note that if $\sum_k \sigma(x^{(k)}) * \sigma(y^{(k)}) = \mathbf{0} \in \mathbb{F}_q^r$ for some $x^{(k)}$'s and $y^{(k)}$'s in $\mathbb{F}_{q^m}$, then $\sum_k x^{(k)} \cdot y^{(k)} = \sum_k \psi(\sigma(x^{(k)}) * \sigma(y^{(k)})) = \psi(\sum_k \sigma(x^{(k)}) * \sigma(y^{(k)})) = \psi(\mathbf{0}) = 0 \in \mathbb{F}_{q^m}$. Using this, it is verified easily that $\mathbf{u}'_0 \in \widehat{C}_1$ would imply $\mathbf{u}_0 \in \widehat{C}$, a contradiction. In conclusion, $0 \in I(\widehat{C}_1)$ and hence, $I(\widehat{C}_1) \neq \emptyset$. We now show $\widehat{t}(C_1) \geq t$. If we call each $j \in \mathcal{I}^*(C)$ a "parent" index, then, using the definition of χ, each of those $\mathbb{F}_{q^m}$-parent indexes can be said to have r $\mathbb{F}_q$-sibling indexes. If $A \subset \mathcal{I}^*(C_1)$ is a non-empty set, then $\beta(A) \subset \mathcal{I}^*(C)$ denotes the set of parent indexes of these siblings. Note that $|\beta(A)| \leq |A|$. Finally, $\alpha(A) \subset \mathcal{I}^*(C_1)$ denotes the set of all siblings of the elements in $\beta(A)$. Note that $A \subset \alpha(A)$. Now let $A \subseteq \mathcal{I}^*(C_1)$ with $|A| = t$. Since $|\beta(A)| \leq t \leq t_0(C)$, there exists $\mathbf{c} \in G$ such that $\pi_0(\mathbf{c}) = 1$ and $\pi_{\beta(A)}(\mathbf{c}) = \mathbf{0}$. Since $\pi'_0(\chi(\mathbf{c})) = 1$, $\pi'_{\alpha(A)}(\chi(\mathbf{c})) = \mathbf{0}$, and $A \subset \alpha(A)$, it follows that $t_0(C_1) \geq t$.

It remains to prove that $r_0(\widehat{C}_1) \le rn - t$. Let $A_1 \subset \mathcal{I}^*(\widehat{C}_1)$ with $|A_1| = t$ be an arbitrary set. Since $A_1 \subset \alpha(A_1)$, it will be sufficient to show that $(0, B_1)$ is a reconstruction-pair for $\widehat{C}_1$, where $B_1 = \mathcal{I}^*(\widehat{C_1}) \setminus \alpha(A_1)$. Write $B = \mathcal{I}^*(\widehat{C}) \setminus \beta(A_1)$. Note that $|B| \ge n - t$. Since $r_0(\widehat{C}) \le n - t$, there exists an $\mathbb{F}_{q^m}$-linear reconstruction function $\rho_{B,0}$ for $\widehat{C}$. We extend the definition of the map ψ as follows: if $\mathbf{x} = (x_0, \mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{F}_q^{1+rn}$, where $x_0 \in \mathbb{F}_q$ and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{F}_q^r$, then $\overline{\psi}(\mathbf{x}) = (x_0, \psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)) \in \mathbb{F}_{q^m}^{n+1}$. Observe that this map is also $\mathbb{F}_q$-linear and that $\overline{\psi}(\chi(\mathbf{c}) * \chi(\mathbf{c}')) = \mathbf{c} * \mathbf{c}'$ for all $\mathbf{c}, \mathbf{c}' \in G$. Moreover if $\pi'_{B_1}(\mathbf{x}) = \pi'_{B_1}(\mathbf{y})$, then observe that $\pi_B(\overline{\psi}(\mathbf{x})) = \pi_B(\overline{\psi}(\mathbf{y}))$. For arbitrary $\mathbf{c}, \mathbf{c}' \in G$, $\rho_{B,0} \circ \pi_B(\mathbf{c} * \mathbf{c}') = \pi_0(\mathbf{c} * \mathbf{c}') = c_0 c'_0 \in \mathbb{F}_q$. Hence, $\rho_{B,0} \circ \pi_B \circ \overline{\psi}(\chi(\mathbf{c}) * \chi(\mathbf{c}')) = c_0 c'_0$. We conclude by this composition and observation above that there exists an $\mathbb{F}_q$-linear map $\rho'_{B_1,0}$ such that $\rho'_{B_1,0} \circ \pi'_{B_1,0}(\chi(\mathbf{c}) * \chi(\mathbf{c}')) = c_0 c'_0$ for all $\mathbf{c}, \mathbf{c}' \in G$. Therefore, since all elements of $\widehat{C}_1$ are of the form $\sum_i \lambda_i \cdot (\chi(\mathbf{c}_i) * \chi(\mathbf{c}'_i))$ with $\lambda_i \in \mathbb{F}_q$ and $\mathbf{c}_i, \mathbf{c}'_i \in G$, and since $\rho'_{B_1,0} \circ \pi'_{B_1,0}$ is an $\mathbb{F}_q$-linear map, it holds that $r_0(\widehat{C}_1) \le rn - t$ as claimed. △

We now present some elementary constructions of multiplication-friendly embeddings.

Theorem 8. *Let $m \ge 2$ be an integer with $q \ge 2m - 2$, then there exists a multiplication-friendly embedding of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ with expansion $2m - 1$.*

Proof. Let $\alpha \in \mathbb{F}_{q^m}$ such that $1, \alpha, \ldots, \alpha^{m-1}$ is a basis of $\mathbb{F}_{q^m}$ as an $\mathbb{F}_q$-vector space. Consider the $\mathbb{F}_q$-vector space $\mathbb{F}_q[X]_{<m}$ of polynomials in $\mathbb{F}_q[X]$ with degree at most $m - 1$. There is an isomorphism of $\mathbb{F}_q$-vector spaces $\phi : \mathbb{F}_q[X]_{<m} \to \mathbb{F}_{q^m}$ given by $f(X) \mapsto f(\alpha)$. Now take $2m - 2$ distinct elements in $\mathbb{F}_q$, $\beta_1, \beta_2, \ldots, \beta_{2m-2}$, and define the map $\xi : \mathbb{F}_q[X]_{<m} \to (\mathbb{F}_q)^{2m-1}$ given by $f(X) \mapsto (f(\beta_1), \ldots, f(\beta_{2m-2}), \mu(f))$ where $\mu(f)$ denotes the coefficient a_{m-1} of X^{m-1} in $f(X)$. Define $\sigma = \xi \circ \phi^{-1}$. For all $x, y \in \mathbb{F}_{q^m}$, we then have that $\sigma(x) = (f(\beta_1), \ldots, f(\beta_{2m-2}), \mu(f))$ and $\sigma(y) = (g(\beta_1), \ldots, g(\beta_{2m-2}), \mu(g))$ where $f(X)$,$g(X) \in \mathbb{F}_q[X]$ are the unique polynomials of degree at most $m - 1$ with $f(\alpha) = x$, $g(\alpha) = y$. We have $\sigma(x) * \sigma(y) = (fg(\beta_1), \ldots, fg(\beta_{2m-2}), \mu(fg))$. Since $f(X) \cdot g(X) \in \mathbb{F}_q[X]$ is of degree at most $2m - 2$, evaluations in $2m - 2$ points of $\mathbb{F}_q$ determine it up to to multiplicative factor (from $\mathbb{F}_q$). This factor is clearly uniquely determined when, in addition, $\mu(fg)$ is taken into account. It follows that $xy = fg(\alpha)$ is determined uniquely by $\sigma(x) * \sigma(y)$, i.e. there exists a function ψ such that $xy = \psi(\sigma(x) * \sigma(y))$ for all $x, y \in \mathbb{F}_{q^m}$. It is not difficult to see that ψ is $\mathbb{F}_q$-linear. △

A construction without any constraint on q and m is presented next. The expansion in this case is quadratic in the degree of the extension. However, for quadratic extensions it is exactly the same as above.

Theorem 9. *There exists a multiplication-friendly embedding of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ with expansion $\binom{m+1}{2}$.*

Proof. Let $\alpha \in \mathbb{F}_{q^m}$ such that $1, \alpha, \ldots, \alpha^{m-1}$ is a basis of $\mathbb{F}_{q^m}$ as an $\mathbb{F}_q$-vector space. Consider the map $\sigma : \mathbb{F}_{q^m} \to (\mathbb{F}_q)^r$ given by $x \mapsto (x_0, \ldots, x_{m-1}, x_0 +$

$x_1, \ldots, x_0 + x_{m-1}, \ldots, x_{m-2} + x_{m-1})$, where $x = \sum_{i=0}^{m-1} x_i \alpha^i$. Given two elements $x, y \in \mathbb{F}_{q^m}$, the coordinates of $\sigma(x) * \sigma(y)$ precisely exhaust all possible expressions $x_i y_i$, as well as all possible expressions $x_i y_i + x_j y_j + x_i y_j + x_j y_i$ for $i \neq j$. Hence, for each pair of indexes (i, j) with $i \neq j$, there exists an $\mathbb{F}_q$-linear map $\phi_{i,j}$ such that $\phi_{i,j}(\sigma(x) * \sigma(y)) = x_i y_j + x_j y_i$. Since $xy = \sum_{k=0}^{2m-2} (\sum_{i+j=k} x_i y_j) \alpha^k = \sum_{i=0}^{m-1} x_i y_i \alpha^{2i} + \sum_{k=0}^{2m-2} (\sum_{i+j=k, i<j} x_i y_j + x_j y_i) \alpha^k$, it follows that there exists an $\mathbb{F}_q$-linear map ψ such that $xy = \psi(\sigma(x) * \sigma(y))$. $\triangle$

Corollary 3. *Let $\mathbb{F}_q$ be a finite field. There exists a multiplication-friendly embedding of $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$ with expansion equal to 3. Moreover, there exists a multiplication-friendly embedding of $\mathbb{F}_{64}$ over $\mathbb{F}_4$ with expansion equal to 5.*

Proof. In the case of quadratic extensions, both multiplication-friendly embeddings give the result. For the second case, we apply Theorem 8. $\triangle$

We are now ready to bound $\widehat{\tau}(q)$ away from zero for all finite fields $\mathbb{F}_q$.

Definition 12. *We define $\nu(q)$ as follows: $\nu(2) = 1/35 \approx 2.86\%$; $\nu(3) = 1/18 \approx 5.56\%$; $\nu(4) = 3/35 \approx 8.57\%$; $\nu(5) = 5/54 \approx 9.26\%$; for q square, $q \geq 49$, $\nu(q) = 1 - \frac{4}{\sqrt{q}-1}$; for the remaining values of q, $\nu(q) = \frac{1}{3}(1 - \frac{4}{q-1})$.*

Theorem 10. *Let $\mathbb{F}_q$ be a finite field. Then $\widehat{\tau}(q) \geq \nu(q)$.*

Proof. If $q \geq 49$ and q is a square, then $\widehat{\tau}(q) \geq (1 - \frac{4}{\sqrt{q}-1})$ by Theorem 6. Using a degree 2 descent from the combination of Theorem 7 and Corollary 3, this immediately yields $\widehat{\tau}(q) \geq \frac{1}{3} \cdot (1 - \frac{4}{q-1})$ if $7 \leq q < 49$, or if $q > 49$ and q is not a square. For $q = 4$, $\mathbb{F}_{64}$ is a degree 3 extension of $\mathbb{F}_4$. Combining Theorem 7, instantiated with the multiplication-friendly mapping from $\mathbb{F}_{64}$ to $(\mathbb{F}_4)^5$ from Corollary 3, with the fact that $\widehat{\tau}(64) \geq \frac{3}{7}$, it follows that $\widehat{\tau}(4) \geq \frac{1}{5} \cdot \frac{3}{7} = \frac{3}{35}$. For $q = 2, 3, 5$ a further degree 2 descent in combination with the results above leads to $\widehat{\tau}(2) \geq \frac{1}{3} \cdot \frac{3}{35} = \frac{1}{35}$, $\widehat{\tau}(3) \geq \frac{1}{3} \cdot \frac{1}{6} = \frac{1}{18}$, and $\widehat{\tau}(5) \geq \frac{1}{3} \cdot \frac{5}{17} = \frac{5}{54}$. $\triangle$

We note that it is possible to further improve these lower bounds especially for small values of q using more advanced techniques from algebraic geometry, as we show in upcoming work [5].

6 Consequences for LSSS with Strong Multiplication

We now state the consequences for LSSS with strong multiplication explicitly.

Definition 13. *Let $\Sigma = (S_0, S_1, \ldots, S_n)$ be an SSS. The (average) information rate $\lambda(\Sigma)$ is defined as $\lambda(\Sigma) = \frac{\sum_{i=1}^{n} H(S_i)}{n \cdot H(S_0)}$. $\mathcal{F} = \{\Sigma_n\}_{n \in N}$ is a family of secret sharing schemes if $N \subset \mathbb{N}$ is an infinite set and for all $n \in N$, Σ_n is a secret sharing scheme with $|\mathcal{P}(\Sigma_n)| = n$. The (average) information rate $\lambda(\mathcal{F})$ of the family $\mathcal{F}$ is the function $\lambda_{\mathcal{F}} : N \longrightarrow \mathbb{R}_{\geq 0}$ with $n \mapsto \lambda(\Sigma_n)$.*

Definition 14. *$\mathcal{F} = \{\Sigma_n\}_{n\in N}$ is a* family of ideal LSSS (over $\mathbb{F}_q$) with strong multiplication *if the following properties hold. $\mathcal{F}$ is a family of secret sharing schemes such that for all $n \in N$, $\Sigma_n = (\mathbb{F}_q, n, e^{(n)}, \mathbf{v}_0^{(n)}, V_1^{(n)}, \ldots, V_n^{(n)})$ is an LSSS. Moreover, for each $n \in N$, Σ_n is "ideal", i.e.,* $\dim V_i^{(n)} = 1$ *for $i = 1, \ldots n,$. Finally, for all $n \in N$, Σ_n has t_n-strong multiplication, where t_n is the maximum integer with that property. The* corruption tolerance $\widehat{t}_{\mathcal{F}}$ *of $\mathcal{F}$ is defined as the function $\widehat{t}_{\mathcal{F}} : N \longrightarrow \mathbb{R}_{\geq 0}$ with $n \mapsto \frac{3t_n}{n-1}$. Such a family is* asymptotically good *if* $\limsup_{n\in N} \widehat{t}_{\mathcal{F}}(n) > 0$, *and* asymptotically bad *otherwise.*

Combining Theorem 10 with Theorem 2, there are the following consequences for strongly multiplicative LSSS.

Theorem 11. *Let $\mathbb{F}_q$ be an arbitrary finite field. There exists an asymptotically good family $\mathcal{F} = \{\Sigma_n\}_{n\in N}$ of ideal LSSS over $\mathbb{F}_q$ with strong multiplication such that* $\lim_{n\to\infty} \widehat{t}_{\mathcal{F}}(n) = \nu(q)$.

Note that over $\mathbb{F}_2$, for example, t is at least a $1/105$-fraction of n, i.e. 0.95% of the players. Also note that by making q large enough, $\nu(q)$ gets arbitrarily close to 1 and, hence, t gets arbitrarily close to $\frac{1}{3}n$.

7 Asymptotically Bad Yet Elementary Schemes

We have shown that a combination of strong methods from algebraic geometry with a dedicated field-descent method leads to asymptotically good schemes over any finite field. We now show an elementary construction that also works over any finite field $\mathbb{F}_q$. However, it is asymptotically bad. Yet it gives t-strong multiplication for $t = \Omega(n/((\log\log n)\log n))$. A combination of results from [7] with replication gives an elementary family with $t = \Omega(\sqrt{n})$, which is much worse. Our construction here consists of applying a combination of Theorems 7 and 9 to Shamir's LSSS over a tower of extension fields of the base field $\mathbb{F}_q$, where the degree of the extension tends to infinity. For every $m > 0$, define $r_m = (q^m)^{\lfloor q^m/2 \rfloor}$. Consider the $[n+1, t]_{r_m}$-Reed-Solomon code C_m with $n+1 = r_m$ and $t = \lfloor \frac{1}{3}(r_m - 2) \rfloor$, i.e. $\Sigma(C_m)$ is a Shamir's LSSS over $\mathbb{F}_{r_m}$ with $r_m - 1$ players and t-strong multiplication. Now apply the construction in Theorem 7 to the codes C_m, using the multiplication-friendly embedding from Theorem 8, and we descend from LSSS over $\mathbb{F}_{r_m}$ to LSSS over $\mathbb{F}_{q^m}$. Using Theorem 9, we descent again from LSSS over $\mathbb{F}_{q^m}$ to LSSS over $\mathbb{F}_q$. Note that the final number of players is now $(r_m - 1)(2\lfloor q^m/2 \rfloor - 1)\binom{m+1}{2}$.

Theorem 12. *Let $\mathbb{F}_q$ be an arbitrary finite field. The above (elementary) construction yields a family $\mathcal{F} = \{\Sigma_n\}_{n\in N}$ of ideal LSSS over $\mathbb{F}_q$ with $t(n)$-strong multiplication, where $t(n) = \Omega(n/((\log\log n)\log n)$.*

Proof. Write $n_m = (r_m - 1)(2\lfloor q^m/2 \rfloor - 1)\binom{m+1}{2}$. The code $\widetilde{C}_m$ constructed as above gives an LSSS Σ_{n_m} for n_m players and t_{n_m}-strong multiplication for $t_{n_m} = \lfloor \frac{1}{3}(r_m - 2) \rfloor$. On the other hand, it is easy to see that $m = O(\log\log n_m)$ and $m \cdot q^m = O(\log n_m)$. The desired result follows. △

8 Upper Bounds on Optimal Corruption Tolerance

So far we have presented asymptotic lower bounds on optimal corruption tolerance. We now turn to (non-asymptotic) upper bounds on corruption tolerance of a code. Using arguments given in [8], it follows easily that $\widehat{\tau}(C) < 1$ for all $C \in C^{\dagger}(\mathbb{F}_q)$ with $\ell(C)$ large enough as a function of q (note that $\widehat{\tau}(C) = 1$ is achievable if $n \leq q$). In Theorem 15 below we improve this bound. The improvement is based on a combination of Theorem 2 with Theorem 13, a more general result for LSSS we prove below. Namely, we lower bound the information rate as a function of the *threshold gap*. Here, the *threshold-gap* of an SSS is defined as the difference between its reconstruction- and privacy-thresholds. A further implication is that in all interesting cases, the threshold gap necessarily grows at least as $\Omega(\log n)$, where n is the number of players, in any family of LSSS over $\mathbb{F}_q$ with positive information rate. Let $\Sigma = (\mathbb{F}_q, n, e, \mathbf{v}_0, V_1, \ldots, V_n)$ be an LSSS over $\mathbb{F}_q$.

Theorem 13. *Set* $\overline{g}(\Sigma) = r(\Sigma) - t(\Sigma)$, *the* threshold gap *of* Σ. *If* $t(\Sigma) \geq 1$ *and* $r(\Sigma) < n$, *then* $\dim \Sigma \geq \frac{n}{\overline{g}(\Sigma)} \cdot \log_q(\frac{n+\overline{g}(\Sigma)+2}{2\overline{g}(\Sigma)})$.

This generalizes a result from [11] where a lower bound in the dimension of any *threshold* LSSS over $\mathbb{F}_2$ is proven. In our result, the threshold gap can be greater than 1 (and q is arbitrary). The proof of Theorem 13 will rely in part on Theorem 14 and Corollary 4 below.

Theorem 14. *Let* $\mathcal{G}$ *be a non-empty collection of subsets of* $\mathcal{P}(\Sigma)$ *such that* $\mathcal{G} \subset \mathcal{A}(\Sigma)$ *and, for any* $A, B \in \mathcal{G}$ *with* $A \neq B$, $A \cup B \in \Gamma(\Sigma)$. *Then,* $\sum_{A \in \mathcal{G}} d_A \geq |\mathcal{G}| \cdot \log_q(|\mathcal{G}|)$, *where* d_A *is the dimension of* V_A *for all* $A \in \mathcal{G}$.

Proof. Our proof uses a lower bound technique from Karchmer and Wigderson [21] (and our claim is essentially a slight generalization of their result). Define $H_1 = \{\phi \in \mathrm{Hom}(\mathbb{F}_q^e, \mathbb{F}_q) : \phi(\mathbf{v}_0) = 1\}$. For all non-empty $A \subset \mathcal{P}$, define $H_{1,A} = H_1 \cap V_A^{\perp}$. Note that, by the characterization from Section 2.2, $A \in \Gamma(\Sigma)$ if and only if $H_{1,A} = \emptyset$. By linear algebra, $|H_1| = q^{e-1}$ and $|H_{1,A}| = q^{e-d_A-1}$ if $A \notin \Gamma(\Sigma)$. Moreover, if $A, B \in \mathcal{G}$, then $A \cup B \in \Gamma(\Sigma)$. Hence, $H_{1,A\cup B} = H_{1,A} \cap H_{1,B} = \emptyset$. Therefore, $|\bigcup_{A \in \mathcal{G}} H_{1,A}| = \sum_{A \in \mathcal{G}} |H_{1,A}|$ So $q^{e-1} = |H_1| \geq |\bigcup_{A \in \mathcal{G}} H_{1,A}| = \sum_{A \in \mathcal{G}} |H_{1,A}| = \sum_{A \in \mathcal{G}} q^{e-d_A-1}$. This gives $\sum_{A \in \mathcal{G}} q^{-d_A} \leq 1$. By the log-sum inequality,[4] $\sum_{A \in \mathcal{G}} d_A \geq |\mathcal{G}| \cdot \log_q(\frac{|\mathcal{G}|}{\sum_{A \in \mathcal{G}} q^{-d_A}}) \geq |\mathcal{G}| \cdot \log_q(|\mathcal{G}|)$. $\triangle$

Definition 15. $\mathcal{G} = \{A_1, \ldots, A_m\}$ *is a* greedy partition *of* $\mathcal{P}(\Sigma)$ *if* m *is a positive integer,* $A_1, \ldots, A_m \subset \mathcal{A}(\Sigma)$, $\bigcup_{i=1}^m A_i = \mathcal{P}(\Sigma)$, $A_i \cap A_j = \emptyset$ $(1 \leq i < j \leq m)$, *and, for* $k = 1, \ldots, m$, A_k *is maximal in* $\mathcal{A}(\Sigma)$ *subject to* $A_k \subset \mathcal{P} \setminus \bigcup_{j=1}^{k-1} A_j$ $(A_0 = \emptyset)$.

[4] The log-sum inequality asserts that for non-negative real numbers $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ $\sum_{i=1}^r a_i \log_q(\frac{a_i}{b_i}) \geq (\sum_{i=1}^r a_i) \log_q \frac{\sum_{i=1}^r a_i}{\sum_{i=1}^r b_i}$.

Note that if $A, B \in \mathcal{G}$, then $A \cup B \in \Gamma(\Sigma)$. If $t(\Sigma) \geq 1$, there exists a greedy partition by induction. Moreover, the size m of a greedy partition can be bounded in terms of $r(\Sigma)$, since any set in the partition has at most $r(\Sigma) - 1$ elements.

Corollary 4. *Suppose* $t(\Sigma) \geq 1$ *and let* $\mathcal{G}$ *be a greedy partition of* $\mathcal{P}(\Sigma)$. *Then* $\dim \Sigma \geq |\mathcal{G}| \cdot \log_q |\mathcal{G}|$. *In particular,* $\dim \Sigma \geq \lceil \frac{n}{r(\Sigma)-1} \rceil \cdot \log_q(\lceil \frac{n}{r(\Sigma)-1} \rceil)$.

Proof (of Theorem 13). We use a dualization technique from [11]. Set $r = r(\Sigma)$, $t = t(\Sigma)$ and $\overline{g} = \overline{g}(\Sigma)$. Note that if Σ^* is an LSSS over $\mathbb{F}_q$ whose access structure is the dual of Σ's, then $\overline{g}(\Sigma^*) = \overline{g}(\Sigma)$. Sort the players $1, \ldots, n$ so that $d_i \leq d_j$ if $i \leq j$. Let Σ_1 be the LSSS restricted to the first $r + 1$ players (this is possible since $r < n$). Clearly $t(\Sigma_1) \geq t(\Sigma)$ and $r(\Sigma_1) \leq r(\Sigma)$, so $\overline{g}(\Sigma_1) \leq \overline{g}(\Sigma)$. There exists an LSSS Σ_1^* over $\mathbb{F}_q$ and defined over the first $r + 1$ players such that $\dim \Sigma_1 = \dim \Sigma_1^*$ and $\Gamma(\Sigma_1^*)$ is the dual access structure to $\Gamma(\Sigma_1)$ (see the remark in Section 2.2). Note that $t(\Sigma_1^*) \geq 1$ and that $r(\Sigma_1^*) = r + 1 - t(\Sigma_1) \leq r + 1 - t = \overline{g} + 1$. By Corollary 4, $\dim \Sigma_1 = \dim \Sigma_1^* \geq \lceil \frac{r+1}{r(\Sigma_1^*)-1} \rceil \cdot \log_q(\lceil \frac{r+1}{r(\Sigma_1^*)-1} \rceil) \geq \lceil \frac{r+1}{\overline{g}} \rceil \cdot \log_q(\lceil \frac{r+1}{\overline{g}} \rceil)$. Because of the sorting of the players, $\dim \Sigma \geq \frac{n}{r+1} \cdot \dim \Sigma_1 \geq \lceil \frac{n}{\overline{g}} \rceil \cdot \log_q(\lceil \frac{r+1}{\overline{g}} \rceil)$. Finally, let Σ^* now be an LSSS over $\mathbb{F}_q$ such that $\dim \Sigma = \dim \Sigma^*$ and $\Gamma(\Sigma^*)$ is the dual access structure to $\Gamma(\Sigma)$ (note that $t(\Sigma^*) \geq 1$ since $r < n$ and $r(\Sigma^*) < n$ since $t \geq 1$). Applying the bound we have just derived, we get $\dim \Sigma^* \geq \lceil \frac{n}{\overline{g}(\Sigma^*)} \rceil \cdot \log_q(\lceil \frac{r(\Sigma^*)+1}{\overline{g}(\Sigma^*)} \rceil)$. But $\bar{g}(\Sigma^*) = \bar{g}$ and $r(\Sigma^*) = n - t$, so $\dim \Sigma = \dim \Sigma^* \geq \lceil \frac{n}{\bar{g}} \rceil \cdot \log_q(\lceil \frac{n-t+1}{\bar{g}} \rceil)$. It is easy to see then that $\dim \Sigma \geq \frac{n}{\bar{g}} \log_q(\frac{n+\bar{g}+2}{2\bar{g}})$. △

Corollary 5. *Let* $\mathcal{F} = \{\Sigma_n\}_{n \in N}$ *be a family of LSSS over* $\mathbb{F}_q$. *If the growth rate of the threshold gap is smaller than logarithmic in the number of players, i.e.,* $\limsup_{n \in N} \frac{\bar{g}(\Sigma_n)}{\log_q n} = 0$, *then the information rate satisfies* $\limsup_{n \in N} \lambda_{\mathcal{F}}(n) = 0$.

Theorem 15. *Let* $C \in C^{\dagger}(\mathbb{F}_q)$. *We have* $\widehat{t}(C) \leq \frac{1}{3} \cdot (n(C) - \frac{1}{2} \cdot \log_q(n(C) + 2))$ *and therefore* $\widehat{\tau}(C) \leq 1 - \frac{\log_q(n(C)+2)-2}{2n(C)-2}$.

Proof. Assume wlog that $\widehat{t}(C)$ is attained for $i = 0$ (i.e., $i_s = 0$, see Definition 8) and write $t = \widehat{t}(C)$. Then $t_0(C) \geq t$ and $r_0(C) \leq n(C) - 2t$, by Theorem 2. Now set $\bar{g} = \bar{g}(\Sigma(C))$ and $n = n(C)$. So, on the one hand, $\widehat{t}(C) \leq \frac{1}{3}(n - \bar{g})$. On the other hand, $\Sigma(C)$ is an "ideal" LSSS. Theorem 13 then implies $n \geq \frac{n}{\bar{g}} \log_q(\frac{n+\bar{g}+2}{2\bar{g}})$. Thus, $\bar{g} \geq \log_q(\frac{n+2}{2\bar{g}}) \geq \log_q(n+2) - \log_q(2\bar{g})$. Then $\bar{g} + \log_q(2\bar{g}) \geq \log_q(n + 2)$, and since $\bar{g} \geq \log_q(2\bar{g})$ for any $\bar{g} \geq 1$, $\bar{g} \geq \frac{1}{2} \log_q(n + 2)$. Combining these facts, the result follows. △

Note that this non-asymptotic upper bound on corruption tolerance does not imply $\widehat{\tau}(q) < 1$.

9 Open Problems

First, our main Theorem 10 implies that the asymptotic optimal corruption tolerance $\widehat{\tau}(q)$ satisfies $\widehat{\tau}(q) > 0$ for all finite fields $\mathbb{F}_q$. The proof of that theorem makes crucial use of strong results from algebraic geometry (namely, good towers of algebraic function fields). Is that essential? Though it is not unlikely that "strong algebraic geometry" is inherent to *strong* lower bounds on $\widehat{\tau}(q)$, is there perhaps a more elementary proof just that $\widehat{\tau}(q) > 0$? Second, it seems likely that the bound from Theorem 15 can be sharpened considerably. Third, it is interesting to improve our lower bounds for $\widehat{\tau}(q)$. We have already noted that in forthcoming work we do so for small values of q, using more advanced methods from algebraic geometry.

Acknowledgements

Ignacio Cascudo was supported by FICYT (project IB-08-147) and Spanish MICINN (project MTM2007-67884-C04-01 and FPU grant AP2005-0836, co-financed by the European Social Fund). Hao Chen was supported by NNSF China Grant 10871068. Ronald Cramer was supported by his NWO VICI project "Mathematical Foundations of Secure Computation." Chaoping Xing was supported by Singapore's NRF Competitive Research Programme (CRP), NRF-CRP 2200703 and the Singapore MoE Tier 2 grant T208B2206.

References

1. Bassa, A., Garcia, A., Stichtenoth, H.: A new tower over cubic finite fields. Moscow Mathematical Journal 8(3), 401–418 (2008)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of STOC 1988, pp. 1–10. ACM Press, New York (1988)
3. Chaum, D., Crépeau, C., Damgaard, I.: Multi-party unconditionally secure protocols. In: Proceedings of STOC 1988, pp. 11–19. ACM Press, New York (1988)
4. Chudnovsky, D.V., Chudnovsky, G.V.: Algebraic complexities and algebraic curves over finite fields. Proceedings of the National Academy of Sciences of the United States of America 84(7), 1739–1743 (1987)
5. Cascudo, I., Cramer, R., Xing, C.: Ongoing work (2009)
6. Chen, H., Cramer, R., de Haan, R., Cascudo Pueyo, I.: Strongly multiplicative ramp schemes from high degree rational points on curves. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 451–470. Springer, Heidelberg (2008)
7. Chen, H., Cramer, R., Goldwasser, S., de Haan, R., Vaikuntanathan, V.: Secure Computation from Random Error Correcting Codes. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 291–310. Springer, Heidelberg (2007)
8. Chen, H., Cramer, R.: Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computation over Small Fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (2006)
9. Cramer, R., Daza, V., Gracia, I., Jimenez Urroz, J., Leander, G., Martí-Farré, J., Padró, C.: On codes, matroids and secure multi-party computation from linear secret sharing schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 327–343. Springer, Heidelberg (2005)
10. Cramer, R., Fehr, S., Stam, M.: Black-Box Secret Sharing from Primitive Sets in Algebraic Number Fields. In: Shoup, V. (ed.) CRYPTO 2005, vol. 3621, pp. 344–360. Springer, Heidelberg (2005)

11. Cramer, R., Fehr, S.: Optimal Black-Box Secret Sharing over Arbitrary Abelian Groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 272–287. Springer, Heidelberg (2002)
12. Cramer, R., Damgaard, I., Dziembowski, S.: On the complexity of verifiable secret sharing and multi-party computation. In: Proceedings of STOC 2000, pp. 325–334. ACM Press, New York (2000)
13. Cramer, R., Damgaard, I., Maurer, U.: General secure multi-party computation from any linear secret sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
14. Damgaard, I., Buus Nielsen, J., Wichs, D.: Isolated Proofs of Knowledge and Isolated Zero Knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)
15. García, A., Stichtenoth, H.: On the asymptotic behavior of some towers of function fields over finite fields. J. Number Theory 61, 248–273 (1996)
16. Goppa, V.D.: Codes on algebraic curves. Soviet Math. Dokl. 24, 170–172 (1981)
17. Huffman, W.G., Pless, V.: Fundamentals of Error Correcting Codes. Cambridge University Press, Cambridge (2003)
18. Ihara, Y.: Some remarks on the number of rational points of algebraic curves over finite fields. J. Fac. Sci. Tokyo 28(3), 721–724 (1981)
19. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer–Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
20. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of 39th STOC, San Diego, Ca, USA, pp. 21–30 (2007)
21. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of the Eigth Annual Structure in Complexity Theory Conference, pp. 102–111. IEEE, Los Alamitos (1993)
22. van Lint, J.H.: Introduction to Coding Theory. Graduate Texts in Mathematics. Springer, Heidelberg (1999)
23. Massey, J.L.: Minimal codewords and secret sharing. In: Proceedings of the 6-th Joint Swedish-Russian Workshop on Information Theory, Molle, Sweden, August 1993, pp. 269–279 (1993)
24. Massey, J.L.: Some applications of coding theory in cryptography. In: Codes and Ciphers: Cryptography and Coding IV, pp. 33–47 (1995)
25. Serre, J.-P.: Rational points on curves over finite fields notes of lectures at Harvard University (1985)
26. Shamir, A.: How to share a secret. Comm. of the ACM 22(11), 612–613 (1979)
27. Stichtenoth, H.: Algebraic function fields and codes. Springer, Heidelberg (1993)
28. Shparlinski, I., Tsfasman, M., Vladuts, S.: Curves with many points and multiplication in finite fields. In: Coding Theory and Algebraic Geometry, pp. 145–169. Springer, Heidelberg (1992)
29. Tsfasman, M., Vladuts, S., Nogin, D.: Algebraic-geometric codes: Basic Notions. AMS, Mathematical Surveys and Monographs, vol. 139 (2007)

The Round Complexity of Verifiable Secret Sharing Revisited

Arpita Patra[1,⋆], Ashish Choudhary[1,⋆⋆],
Tal Rabin[2], and C. Pandu Rangan[1,⋆⋆⋆]

[1] Dept of Computer Science and Engineering
IIT Madras, Chennai, India 600036
arpitapatra10@gmail.com, partho_31@yahoo.co.in, prangan55@gmail.com
[2] IBM T.J. Watson Research Center
talr@us.ibm.com

Abstract. The round complexity of interactive protocols is one of their most important complexity measures. In this work we prove that existing lower bounds for the round complexity of VSS can be circumvented by introducing a negligible probability of error in the reconstruction phase. Previous results show matching lower and upper bounds of *three* rounds for VSS, with $n = 3t + 1$, where the reconstruction of the secrets always succeeds, i.e. with probability 1. In contrast we show that with a negligible probability of error in the reconstruction phase:

1. There exists an efficient 2-round VSS protocol for $n = 3t + 1$. If we assume that the adversary is non-rushing then we can achieve a 1-round reconstruction phase.
2. There exists an efficient 1-round VSS for $t = 1$ and $n > 3$.
3. We prove that our results are optimal both in resilience and number of sharing rounds by showing:
   (a) There does not exist a 2-round WSS[1] (and hence VSS) for $n \leq 3t$.
   (b) There does not exist a 1-round VSS protocol for $t \geq 2$ and $n \geq 4$.

1 Introduction

Verifiable Secret Sharing (VSS) [3] is a fundamental building block for many distributed cryptographic tasks. VSS is a two phase protocol (Sharing and Reconstruction) carried out among n parties in the presence of an adversary who can corrupt up to t parties. Informally, the goal of the VSS protocol is to share a secret, s, among the n parties during the sharing phase in a way that would

⋆ Financial Support from Microsoft Research India Acknowledged.
⋆⋆ Financial Support from Infosys Technology India Acknowledged.
⋆⋆⋆ Work Supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation Sponsored by Department of Information Technology, Government of India.
[1] WSS is a weaker notion of VSS.

later allow for a unique reconstruction of this secret in the reconstruction phase, while preserving the secrecy of s until the reconstruction phase.

Due to the central importance of VSS in the context of many cryptographic protocols such as multiparty computation, Byzantine agreement, etc, the problem has drawn much attention over the years (e.g. [1,2,4,5,6,7,8,12,18]) and many aspects of the problem have been studied. Round complexity is one of the most important complexity measures of interactive protocols. The study of the round complexity of VSS in the information theoretic security setting, i.e. under the assumption of a *computationally unbounded adversary*, was initiated by Gennaro et al. [11]. Their investigation was conducted under the assumption that the protocols are error-free. They refer to the round complexity of VSS as the number of rounds in the sharing phase and prove that a 3-round error-free VSS is possible only if $n \geq 3t+1$, and match it with an inefficient upper bound. Fitzi et al. [10] show an optimal efficient 3-round VSS protocol in this setting. The protocol of Fitzi et al. used the broadcast channel in more than one round of the sharing phase and Katz et al. [14] showed how to achieve the same result while using a single round of broadcast. The lower bound from [11] (and the matching upper bounds) consider *error-free* VSS, where the VSS properties are satisfied without any probability of error.

In this work we investigate the question of whether the lower bounds for the round complexity of VSS can be overcome by introducing a negligible probability of error.

Our Results: We prove that existing lower bounds for the round complexity of VSS can be circumvented by introducing a negligible probability of error in the reconstruction phase. Specifically, we show that:

1. There exists an efficient 2-round VSS protocol for $n = 3t + 1$. This protocol has a 2-round reconstruction phase. If we assume that the adversary is non-rushing then we can achieve a 1-round reconstruction phase. A rushing adversary can wait to hear the incoming messages in a given round prior to sending out its own messages.

   This matches the sharing phase round complexity of the best known protocols in the computational setting [9,16] with no set-up assumptions (but note that these protocols use a one round reconstruction phase).
2. There exists an efficient 1-round VSS for $t = 1$ and $n \geq 4$.
3. We prove that our results are optimal both in resilience and number of sharing rounds by proving:
   (a) There does not exist a 2-round WSS (and hence VSS) for $n \leq 3t$.
   (b) There is no 1-round VSS protocol for $t \geq 2$ and $n \geq 4$[2].

Our protocols also achieve the design optimization of Katz et al. [14] and use a single round of broadcast in the sharing phase and no broadcasts at all in the reconstruction phase.

To achieve our goal of constructing a VSS protocol, we follow the structure of [17,18], where we first design a Weak Secret Sharing (WSS) protocol and

[2] We note that there exists a 1-round WSS protocol with $n > 3t$ (see Appendix A).

then use it as a building block for VSS. Informally WSS is a primitive which satisfies the same properties as VSS except for the commitment property. VSS has a *strong commitment*, which requires that at the end of the sharing, there is a fixed value s^* and that the honest parties output this value in the reconstruction phase. In contrast, WSS has a *weaker commitment* property which requires that at the end of the reconstruction phase, the honest parties output s^* or NULL. The novelty of our protocol is in the specific design of the WSS component and the way we use it to build the round optimal VSS.

On the Definition of Round Complexity of VSS: As we have stated earlier, the common definition for the round complexity of VSS is the number of rounds in the sharing phase. This is a natural definition for the perfect (i.e., zero error) setting, as the reconstruction can always be done in one round (by having all parties reveal their complete view generated at the end of sharing phase). However, in our protocols we have a reconstruction phase that cannot be collapsed into a single round. This indicates that a different definition for the round complexity of VSS may be needed, which is the total number of rounds in the sharing plus the number of rounds in the reconstruction. Both the previous VSS results [10,11,14] and our result exhibit a VSS with a *total* of four rounds[3]. This introduces the question of what is the lower bound on the total number of rounds for VSS.

2 Preliminaries

We follow the network model of [11,18]. Specifically, we consider a setting with n parties $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ that are pairwise connected by a private and authenticated channel. We further assume that all parties have access to a common broadcast channel and there exists a *malicious, computationally unbounded adversary* $\mathcal{A}_t$, that can corrupt up to t parties, out of n parties. The adversary controls and coordinates the actions of the corrupted/faulty parties. We further allow the adversary to be rushing, i.e. in every round of communication it can wait to hear the messages of the honest parties before sending his own messages. For simplicity we describe our protocols for a static adversary, who corrupts all the parties at the beginning of the protocol. However, our results also hold for a stronger adaptive adversary. Given a security parameter k, we assume that the protocols operate with values from a finite field $\mathbb{F} = GF(q)$, where $q = 2^k$. Thus, each element of $\mathbb{F}$ can be represented by k bits. Moreover, without loss of generality, we assume that $n = \text{poly}(k)$. The error probability of our protocols will be $2^{-\Omega(k)}$. We say that our protocols are efficient if the communication and computation of the parties are polynomial in the security parameter k. All the protocols presented in this paper perform computation and communication

[3] As the total number of rounds in both protocols is the same, the question of which protocol to use depends on the application. For applications where there is a need of more efficiency during the sharing, i.e. fewer number of rounds, the two round sharing protocol should be used.

which are poly(k). We assume the system to be synchronous. Therefore the protocols operate in a sequence of rounds, where in each round, a party performs some local computation, sends new messages to the other parties through the private channels and broadcasts some information over the broadcast channel, then it receives the messages that were sent by the other parties in this round on the private and broadcast channels.

2.1 Verifiable Secret Sharing (VSS)

We now present the definition of VSS [3]. In a VSS protocol there is a distinguished party $D \in \mathcal{P}$, that holds an input $s \in \mathbb{F}$ referred to as the secret. The protocol consists of two phases, a sharing phase and a reconstruction phase. We call an n party protocol with adversary $\mathcal{A}_t$ an (n,t)-*VSS* protocol if it satisfies the following conditions for dealer D holding secret s :

Secrecy. If D is honest then the adversary's view during the sharing phase reveals no information on s.[4] More formally, the adversary's view is *identically distributed* for all different values of s.

Correctness. If D is honest then the honest parties output s at the end of the reconstruction phase.

Strong Commitment. If D is *corrupted*, then at the end of the sharing phase there is a value $s^* \in \mathbb{F} \cup \{NULL\}$, such that at the end of the reconstruction phase all honest parties output s^*.

NOTE: This definition is equivalent to saying that $s^* \in \mathbb{F}$, by fixing a default value in $\mathbb{F}$, which may be output in case the reconstruction ends with a NULL. However, we prefer this presentation of the definition as to distinguish it from a stronger definition of VSS [13,11]. The stronger definition also requires that at the end of the sharing there is a commitment to an actual value in $\mathbb{F}$, i.e. the dealer cannot commit to NULL, and furthermore that all parties hold a share of this actual value. Thus, using the above definition points to the fact that NULL is a possible value, instead of setting it to a default value in $\mathbb{F}$.

Protocols that do not satisfy the stronger VSS definition are not suitable for use in multiparty computations. The protocols in this paper satisfy the standard VSS definition, which leave the open question of whether a 2-round VSS protocol can be designed that satisfies the stronger definition. However, when examining the round complexity of VSS as a stand alone application, the above definition is sufficient and was used in [11] (with the variation $s^* \in \mathbb{F}$) to prove the lower bounds.

VSS in External Dealer Model: In the external dealer model, the system is assumed to consist of a dealer and n parties. The dealer is considered as an external party. Moreover, the adversary $\mathcal{A}_t$ is allowed to corrupt D and up to t *additional* parties. We stress that all the protocols and lower bounds presented in this paper will work for this model as well.

[4] If D is corrupted, then s will be known to the adversary. In such a case, the secrecy property does not apply.

2.2 Weak Secret Sharing (WSS)

In order to construct our VSS protocol we use another form of secret sharing called Weak Secret Sharing (WSS) [17,18]. The setting is the same as for the VSS and the definition satisfies the Secrecy and Correctness properties. However, we relax the Commitment property as follows:

Weak Commitment. If D is faulty then at the end of the sharing phase there is a value $s^* \in \mathbb{F} \cup \{NULL\}$ such that at the end of the reconstruction phase, each honest party will output either s^* or NULL.

Notice that it is not required that all honest parties output the same value, i.e. some may output s^* and some may output NULL. The above definition is standard and follows many of the existing definitions [14,17,18].

2.3 Statistical VSS and Statistical WSS

We say that a VSS (WSS) protocol is a $(1-\epsilon)$ statsitical VSS (WSS) if it achieves correctness and strong (weak) commitment with probability $1-\epsilon$, where given a security parameter k we have that $\epsilon = 2^{-\Omega(k)}$. Note that we assume secrecy to be perfect[5].

3 Statistical-WSS, 2-Round Sharing, $n = 3t + 1$

In this section we present our 2-round share, 2-round reconstruct statistical-WSS protocol with $n = 3t+1$. The protocol appears in Figure 1. For ease of exposition, we describe our protocol using multiple rounds of broadcast. We follow this with a brief description on how to modity the protocol to a variation that uses a single round of broadcast.

NOTE: Following the notation of [11], whenever we say that dealer is disqualified during the sharing phase of WSS/VSS, we mean to say that all honest parties accept the sharing of NULL (or a default value from $\mathbb{F}$) as the dealer's secret.

Before we turn to our proofs we draw the readers attention to the following interesting points that enable us to achieve the final result. The bi-variate polynomial $F(x, y)$ (defined by D) has a tweak, the x variable is of degree $nk + 1$, which results in the polynomials $f_i(x)$ being of degree $nk + 1$ (where as this degree is typically t in other protocols). We further create a situation where these polynomials never need to be reconstructed and thus the parties need not hold large number of points on the polynomials to interpolate them. These two properties put together, enable us to give each party many evaluation points and values on these polynomials and to further allow them to expose a portion of them without exposing the underlying polynomial. In addition, we adapt an interesting technique from Tompa and Woll [20] and use *secret* evaluation points.

[5] We conjecture that the lower bounds in this paper hold also for the case when the secrecy is statistical.

Protocol WSS

Sharing Phase

Local Computations: D does the following:

1. Picks a random bivariate polynomial $F(x,y)$ over $\mathbb{F}$ of degree t in the variable y and degree $nk+1$ in the variable x, such that $F(0,0)=s$.
2. Defines $f_i(x)=F(x,i)$ for $1 \le i \le n$.
3. Picks random polynomials $r_i(x)$ over $\mathbb{F}$, $deg(r_i(x))=nk+1$ for $1 \le i \le n$.
4. nk random, non-zero, distinct elements from $\mathbb{F}$, denoted by $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,k}$ for $1 \le i \le n$.

Round 1: D sends to party P_i:

- The polynomials $f_i(x), r_i(x)$. Let $f_i(0)$ be P_i's share of D's secret s.
- The random evaluation points $\alpha_{i,\ell}$ for $1 \le \ell \le k$.
- $a_{j,i,\ell} = f_j(\alpha_{i,\ell})$ and $b_{j,i,\ell} = r_j(\alpha_{i,\ell})$ for $1 \le \ell \le k,\ \ 1 \le j \le n$.

Round 2: Party P_i broadcasts the following:

- A random non-zero value c_i and polynomial $g_i(x) = f_i(x) + c_i r_i(x)$, $deg(g_i(x)) = nk+1$.[a]
- For a random subset of indices $\ell_1, ..., \ell_{\frac{k}{2}}$, the evaluation points $\alpha_{i,\ell_1}, ..., \alpha_{i,\ell_{\frac{k}{2}}}$ and $a_{j,i,\ell_1}, ..., a_{j,i,\ell_{\frac{k}{2}}}$ and $b_{j,i,\ell_1}, ..., b_{j,i,\ell_{\frac{k}{2}}}$ for $1 \le j \le n$.

Local Computation: For all parties:

1. Party P_i is *accepted* by party P_j if $a_{i,j,\ell} + c_i b_{i,j,\ell} = g_i(\alpha_{j,\ell})$ for all ℓ in the set of indices broadcasted by P_j in **Round 2**.
2. Initiate the set $SH = \emptyset$. Place P_i in SH if it is accepted by at least $2t+1$ parties.
3. If $|SH| \le 2t$ disqualify dealer D. Note that SH computed by all honest parties are identical.

Reconstruction Phase, 2-rounds:

Round 1: Each P_i in SH broadcasts $f_i(x)$, $deg(f_i(x)) = nk+1$.
Round 2: Each $P_j \in \mathcal{P}$ broadcasts all the evaluation points $\alpha_{j,\ell}$ which were not broadcasted in the sharing phase and $a_{i,j,\ell}$ corresponding to those indices, for $i = 1, \ldots, n$.

Local Computation: For all parties:

1. Party $P_i \in SH$ is *re-accepted* by $P_j \in \mathcal{P}$ if for one of the newly revealed points it holds that $a_{i,j,\ell} = f_i(\alpha_{j,\ell})$.
2. Initiate the set $REC = \emptyset$. Place P_i in REC if it is re-accepted by at least $t+1$ parties. If the shares of the parties in REC interpolate to a t degree polynomial $g(y)$ then output $s = g(0)$. Otherwise output $NULL$.

[a] When ever we say that a party broadcasts a polynomial of a certain degree we assume that if this is not done then the party is disqualified.

Fig. 1. (2-Round Share, 2-Round Reconstruct) Statistical-WSS, $n = 3t+1$

The fact that we can expose points on the high degree polynomials and that the evaluation points are secret, facilitates the cut-and-choose proof, carried out by the parties in Round 2. It should be noted that if we allow rushing, then a cheating prover may try to foil the cut-and-choose proof during the sharing phase. However, surprisingly we show that this proof is sufficient for our needs and that we can deal with such faulty parties in the reconstruction phase.

Lemma 1. *Protocol WSS satisfies the* $(1-\epsilon)$*-correctness property.*

Proof: It is easy to see that if D is honest, then every honest party P_i is present in SH as well as in REC. Given that all honest parties are present in SH the dealer will not be disqualified during the sharing phase. In order to show that the correct secret is reconstructed, we prove that if a faulty P_i broadcast a polynomial $\bar{f}_i(x) \neq f_i(x)$, then with high probability P_i will not be added to REC. In order for a faulty P_i to be included in REC, it needs to be re-accepted by $t+1$ parties and thus by at least one honest party. The polynomial $\bar{f}_i(x)$ can agree with $f_i(x)$ in at most $nk+1$ evaluation points. Without knowing the secret evaluation points of an honest party, say P_j, the probability that P_i will be re-accepted by P_j is at most $\frac{nk}{|\mathbb{F}|}$. Thus, the probability that any faulty party is in REC is $\frac{(nk)(2t+1)(t)}{|\mathbb{F}|} \approx 2^{-\Omega(k)}$. Hence with very high probability, the parties will reconstruct $s = f(0)$, which is D's secret. □

Note that in the previous proof we did not claim, and in fact cannot claim, that there are no faulty parties in SH. As we allow the adversary to be rushing, it can cause faulty parties, i.e. parties that have broadcasted inconsistent polynomials (during the second round of the sharing phase), to be included in this set. This is done by waiting to hear the evaluation points of the honest parties (in the second round of the sharing phase). However, this does not affect the result of the reconstruction because the parties in SH broadcast their polynomials in the first round while the secret evaluation points of the parties are revealed only in the second round of the reconstruction.

Lemma 2. *Protocol WSS satisfies the* $(1-\epsilon)$*-weak commitment property.*

Proof: To prove this lemma we need to show that in case that a faulty D was not disqualified, i.e. $|SH| \geq 2t+1$, then with high probability, all the honest parties P_i that are in SH are also present in REC. If we prove this then the lemma follows immediately; we set D's committed secrets s^* to be the constant term of the polynomial, which is defined by the interpolation of the shares of the honest parties in SH (note that s^* may be $NULL$). As we require that shares of all the parties in REC define a polynomial of degree t, then either the value s^* or $NULL$ will be reconstructed.

In order for an honest P_i to be in SH and not in REC it must be the case that at least $2t+1$ parties should have accepted P_i in the sharing phase but at most t of them re-accepted it in the reconstruction phase. This means that there is at least one honest P_j who accepted P_i but did not re-accept it. This implies that the data (evaluation points and values) that P_j exposed in the sharing phase satisfies the polynomial $g_i(x)$ that P_i broadcasted during the sharing phase, but on the other hand, out of the remaining evaluation points that are used by P_j in the reconstruction phase, none satisfy the polynomial $f_i(x)$ produced by P_i. That is, for the selected $\frac{k}{2}$ indices $\ell_1, ..., \ell_{\frac{k}{2}}$, it holds that $a_{i,j,\ell} + c_i b_{i,j,\ell} = g_i(\alpha_{j,\ell})$ for all ℓ in the set of indices $\{\ell_1, ..., \ell_{\frac{k}{2}}\}$ and $f_i(\alpha_{j,\ell}) \neq a_{i,j,\ell}$ for all ℓ in the *remaining* set of indices. Notice that P_i chooses c_i independently of the values given by D. Also, P_j chooses the $\frac{k}{2}$ indices randomly out of k indices. So the probability that the above event happens is $\frac{1}{\binom{k}{k/2}} \approx 2^{-\Omega(k)}$, which is negligible. This shows that with high probability all honest parties from SH will be included in REC, thus proving our lemma. □

Lemma 3. *Protocol WSS satisfies perfect secrecy.*

Proof: The secrecy has to be argued when D is honest. For simplicity, assume that first t parties are corrupted. So in **Round 1** of the **Sharing Phase**, the adversary will know the polynomials $f_1(x), \ldots, f_t(x), r_1(x), \ldots, r_t(x)$ and kt points on $f_i(x)$ and $r_i(x)$ for $t+1 \leq i \leq n$. In **Round 2** of the **Sharing Phase**, the adversary learns $\frac{k}{2}(2t+1)$ additional points on $f_i(x)$ and $r_i(x)$ for $t+1 \leq i \leq n$. So in total the adversary will know $kt + \frac{k}{2}(2t+1)$ points on each of $f_i(x)$ and $r_i(x)$ for $t+1 \leq i \leq n$ which is less than the degree of the polynomials $(nk+1)$. Thus, the constant term of the polynomials $f_i(x)$ for $t+1 \leq i \leq n$ are information theoretically secure in the **Sharing Phase**, which further implies information theoretic security for s. □

Theorem 1. *There exists an efficient 2-round share, 2-round reconstruct* $(3t+1, t)$ *statistical-WSS protocol.*

Proof: Protocol WSS presented here achieves $1-\epsilon$-correctness, $1-\epsilon$-weak commitment and perfect secrecy. This follows from Lemma 1, 2 and 3. □

Important Note: There is another interesting way to interpret the computation done in the Protocol WSS. We may view this as D sharing a t degree polynomial $g(y)$ using protocol WSS. For this, D selects the bivariate polynomial $F(x, y)$ as in protocol WSS, such that $F(0, y) = g(y)$. The polynomial $g(y)$ is the polynomial that D used to share the secret $g(0) = F(0, 0) = s$. The polynomial $g(y)$ is not random but only preserves the secrecy of the constant term. Yet, this distribution of polynomials is sufficient to provide the secrecy requirements needed by our protocols.

Statistical WSS with One Round of Reconstruction: It is interesting to note that if we restrict the adversary to a non-rushing adversary then the two rounds of the reconstruction phase can be collapsed into a single round. The two rounds are needed in order to force the adversary to commit to the polynomials $f_i(x)$ of the faulty parties prior to seeing the evaluation points, as this knowledge can enable the adversary to publish a polynomial that is re-accepted by the honest parties, which would violate the correctness of the protocol. However, if the adversary is non-rushing then this property is achieved via the synchronicity of the step. We state this in the following theorem:

Theorem 2. *If the adversary is non-rushing then there exists an efficient 2-round share 1-round reconstruct* $(3t+1, t)$ *statistical-WSS protocol.*

Statistical WSS with One Round of Broadcast: We now show how the protocol in Fig. 1 can be modified, so that it uses *only one* round of broadcast. Specifically, we modify the **Reconstruction Phase**, so that it requires no broadcast.

Reconstruction Phase, 2-rounds

Round 1: Each P_i in SH privately sends $f_i(x)$, $deg(f_i(x)) = nk+1$ to every other party.

Round 2: Each $P_j \in \mathcal{P}$ privately sends all the evaluation points $\alpha_{j,\ell}$ which were not broadcasted in the sharing phase and $a_{i,j,\ell}$ for those indices, to all other parties.

Local Computation: For all parties it is the same as in the Protocol WSS.

This modified version of WSS preserves the $(1-\epsilon)$-correctness and perfect secrecy properties. It will also satisfy $(1-\epsilon)$-weak commitment, but *without agreement.* That is, some honest party(ies) may output the committed secret s^* while some other may output $NULL$.

4 Statistical-VSS, 2-Round Sharing, $n = 3t+1$

We now design a 2-round share, 2-round reconstruct $(3t+1, t)$ statistical-VSS protocol. We follow the general idea of [1,11,10,14] of sharing the secret s with a symmetric bivariate polynomial $F(x, y)$ where each party P_i gets the univariate polynomial $f_i(y) = F(i, y)$ and his share is $f_i(0)$. The next step is for every pair of parties to verify that they have received the correct values from the dealer. However, as we have only one more round available we cannot depend on D to resolve conflicts in a third round. Thus, instead of doing the verification point wise we carry out the verification on polynomials. More specifically, party P_i initiates an execution of the WSS protoocol in the first round, to share a random polynomial $g_i(y)$. In the second round, P_i broadcasts the masked polynomial $h_i(y) = f_i(y) + g_i(y)$, while every other party broadcasts the corresponding point on $h_i(y)$. In fact, this verification can be viewed as an extension of the round reducing technique of pad sharing for a single value given in [11], to the sharing of polynomial, which is used as a pad for the verification of a polynomial. The VSS protocol appears in Figure 2.

Lemma 4. *Protocol VSS satisfies* $(1-\epsilon)$*-correctness property.*

PROOF: A simple examination of the Protocol VSS and the properties of Protocol WSS reveal that all honest parties will be in VSS-SH and thus an honest D is not disqualified during the sharing phase. To prove this lemma we need to show that when D is honest, then very with high probability, for all faulty parties P_j in VSS-SH the following holds: if at the end of WSS^{P_j}, the fixed (weak committed)

Protocol VSS

Sharing Phase

Round 1:

– D selects a random symmetric bivariate polynomial $F(x, y)$ over $\mathbb{F}$ of degree t in each variable such that $F(0, 0) = s$ and sends the polynomial $f_i(y) = F(i, y)$ to P_i.
– Party P_i initiates Round 1 of the WSS protocol to share a random t degree polynomial $g_i(y)$. Denote this execution by WSS^{P_i}.

Round 2:

– Party P_i broadcasts the polynomial $h_i(y) = f_i(y) + g_i(y)$, $deg(h_i(y)) = t$, and values $a_{ji} = f_i(j) + g_j(i) = f_j(i) + g_j(i)$, for $1 \leq j \leq n$.
– Execute Round 2 of the sharing phase of each WSS^{P_i}. Let SH_i denote the set SH from this execution.

Local Computation: For all parties

1. Party P_i *is accepted* by party P_j if $h_i(j) = a_{ij}$.
2. Let $Accept_i$ denote the set of parties that accepted P_i.
3. Create the set VSS-SH. Place P_i in VSS-SH if $|Accept_i| \geq 2t + 1$.
4. Remove P_i from VSS-SH if $|\text{VSS-}SH \cap Accept_i \cap SH_i| \leq 2t$. Repeat, until no more parties can be removed.
5. If $|\text{VSS-}SH| \leq 2t$ then disqualify D.

Reconstruction Phase, 2-rounds:

For all P_i in VSS-SH, execute the 2-round reconstruction phase of WSS^{P_i}. If the output of the execution is not NULL then let $g_i(y)$ be the output from this execution.

Local Computation (for each party)

1. Initialize $REC = \text{VSS-}SH$.
2. Remove P_i from REC if the output of WSS^{P_i} is $NULL$.
3. For each $P_i \in REC$, define its share as $f_i(0) = h_i(0) - g_i(0)$.
4. If the shares of the parties in REC define a unique polynomial $f(x)$ of degree t then output $f(0)$, otherwise output NULL.

Fig. 2. (2-Round Share, 2-Round Reconstruct) Statistical VSS, $n = 3t + 1$

value is not NULL and the shared polynomial is $g_j(y)$, then $h_j(y) - g_j(y)$ is in fact polynomial $f_j(y)$, received by P_j from D. If we prove this, then the lemma follows immediately because, a faulty P_j in VSS-SH whose reconstruction of WSS^{P_j} fails is removed from REC. Furthermore, with high probability, a sufficient number of shares belonging to the parties in REC will be reconstructed successfully (due to the properties of WSS) and thus the correct secret of D will be reconstructed.

What this implies is that we cannot guarantee that all parties in VSS-SH are honest. But we can ensure that if they eventually remain in REC then they have shared the proper values. And this is sufficient to guarantee the correctness of the protocol. We now proceed to prove this claim.

Since P_j is present in VSS-SH, we know that $|Accept_j \cap SH_j| \geq 2t+1$. This means that there are $t+1$ honest parties in this set. By the properties of WSS, this set of honest parties define the polynomial $g_j(y)$ which P_j is committed to, at the end of the sharing phase of WSS^{P_j}. We now examine the polynomial $h_j(y) - g_j(y)$ and show that it is equal to $f_j(y)$. The set of $(t+1)$ honest parties in $(Accept_j \cap SH_j)$ verified that the sum of the share $f_i(j) = f_j(i)$ (which they received from D) and $g_j(i)$ (which they received from P_j), in fact lie on the polynomial $h_j(y)$. Moreover, the set of $t+1$ shares, corresponding to these honest parties define the polynomial $f_j(y)$. Thus, $h_j(y) - g_j(y) = f_j(y)$. □

Lemma 5. *Protocol VSS satisfies* $(1-\epsilon)$*-strong commitment property.*

Proof: If D is corrupted and does not get disqualified during the sharing phase, then VSS-SH is fixed at the end of sharing phase. Since VSS-$SH \geq 2t+1$, it contains a set $\mathcal{H}$ of honest parties of size at least $t+1$. If $f_j(y)$'s corresponding to the parties $\mathcal{H}$ define a unique symmetric bivariate polynomial $F^*(x, y)$ of degree t in x and y, then D's committed secret is $s^* = F^*(0, 0)$. Otherwise, $s^* = \text{NULL}$. We show that in the reconstruction phase s^*. will be reconstructed.

It is easy to see that due to the WSS reconstruction properties, with high probability, all the honest parties in $\mathcal{H} \subseteq$ VSS-SH will also be present in REC. We now divide our proof into two cases: (a) $s^* \neq \text{NULL}$: the proof for this case follows from the proof of Lemma 4 as this case is indistinguishable from the case when D is honest. (b) $s^* = \text{NULL}$: As $\mathcal{H} \subseteq REC$, during Step 4 of the reconstruction phase all parties will output NULL which is equal to s^*. □

Lemma 6. *Protocol VSS satisfies perfect secrecy.*

Proof: This proof is similar to the entropy based argument, used to prove the secrecy of 3 round perfect VSS protocol of [10]. □

Theorem 3. *There exists an efficient 2-round share, 2-round reconstruct* $(3t+1, t)$ *statistical-VSS protocol.*

As the reconstruction phase of the VSS protocol is simply the reconstruction phase of the WSS, we claim here as well, that the reconstruction phase can be collapsed into one round against a non-rushing adversary.

Theorem 4. *If the adversary is non-rushing then there exists an efficient 2-round share 1-round reconstruct* $(3t+1, t)$ *statistical-VSS protocol.*

We stress that in Protocol VSS, D can commit $NULL$ at the end of the sharing phase. This makes Protocol VSS unsuitable for Multiparty Computation. It is an interesting problem to see whether there exists an efficient 2-round share, $(3t+1, t)$ statistical VSS protocol, which satisfies the stronger definition of VSS [13,11], given in Section 2. In fact, if such a sharing exists then it would also imply that there is a one round reconstruction, as error correction can be used to interpolate the secret.

Statistical VSS with One Round of Broadcast: We now explain how Protocol VSS can be modified, so that the broadcast channel is used in only one round throughout the protocol, namely in the second round of the sharing phase. The reconstruction phase of the VSS protocol is simply the reconstruction phase of the WSS protocol. Moreover, in the previous section, we have seen how Protocol WSS can be modified, so as to have only one round of broadcast. Thus, if we can argue that the modified WSS is sufficient for the reconstruction of VSS, then we have a VSS protocol that does not use broadcast in the reconstruction phase. Examining the proof of the VSS protocol, we see that it is not mandatory that the set of shares, which the honest parties use in reconstruction is identical, but rather that it has a large enough intersection. As the shares of the honest parties provide this guarantee, it is irrelevant which shares of the faulty parties are included in the computation. Thus, by using the modified statistical WSS, we get a statistical VSS, with only one round of broadcast.

5 Lower Bounds

5.1 Lower Bound for 2-Round Statistical-VSS, $n \leq 3t$

We now prove the optimality of our 2-round share $(3t+1, t)$ statistical VSS protocol, with respect to the resiliency.

Theorem 5. *There is no 2-round share (n, t)-statistical-VSS protocol with $n \leq 3t$, irrespective of the number of rounds in the reconstruction phase.*

In fact we prove the following stronger result from which the above theorem follows immediately.

Theorem 6. *There is no 2-round share (n, t)-statistical-WSS protocol with $n \leq 3t$, irrespective of the number of rounds in the reconstruction phase.*

To prove the above theorem, we use standard player partitioning arguments and prove the following:

Lemma 7. *There is no 2-round share $(3, 1)$-statistical-WSS protocol, irrespective of the number of rounds in the reconstruction phase.*

Before proceeding to prove the above lemma, we recall the following result:

Lemma 8 ([11]). *Let ψ be any r-round protocol, where $r \geq 2$. Then there exists an r-round protocol $\bar{\psi}$ with the same number of parties and same properties (as ψ), such that all messages in rounds $2, \ldots, r$ of $\bar{\psi}$ are broadcast messages.*

We now prove Lemma 7 by contradiction. Let Π be a 2-round share $(3, 1)$ statistical WSS protocol, having $r \geq 1$ rounds in the reconstruction phase. Let the three parties in Π be P_1, P_2 and P_3, where P_1 is the dealer (D). We prove the lemma by constructing a sequence of executions of Π which allows to show that Π violates the $(1-\epsilon)$-weak commitment property. From Lemma 8, we can assume that in protocol Π, the private communication is done only in the first round, while in the remaining rounds, parties use only broadcast. The broadcasts done by P_2 and P_3 during first round of sharing phase will be independent of the messages received from D and hence can be ignored. Similarly, due to the secrecy property, the broadcast done by D during first round of sharing will be independent of the secret and can be ignored. Moreover, during first round of sharing phase, the private communication done between P_2, P_3 will be independent of the secret. Also the private communication from P_2 to P_1 and from P_3 to P_1 will be independent of the secret.

We first consider the following two executions of Π, where D is honest:

1. In execution E_s, D shares the secret s. In the first round of the sharing phase, D defines the shares s_1, s_2, s_3 and sends them to P_1, P_2 and P_3 respectively. In the second round of sharing, the parties broadcast B_1, B_2 and B_3 respectively. During the reconstruction phase, the parties broadcast messages $C_{1,1}, C_{2,1}$ and $C_{3,1}$ respectively in the first round. For $i = 2, \ldots, r$, in round i of the reconstruction phase, the parties broadcast the messages $C_{1,i}, C_{2,i}$ and $C_{3,i}$ respectively. As D is honest, due to correctness property of Π, the honest parties need to output s at the end of the reconstruction phase.
2. In execution E_{s^*}, D shares the secret s^* and defines the shares $\bar{s}_1, s_2$ and $\bar{s}_3$ respectively and gives them to P_1, P_2 and P_3 respectively. Note that due to the secrecy property of Π, such a sharing always exists. Given different randomness, we can have the broadcast messages in round two of the sharing phase be identical to B_1, B_2 and B_3 respectively.[6] The broadcasts in the reconstruction phase are as follows: note that P_2's view is identical to its view in E_s up to this step and thus the first round messages of P_2 in the reconstruction phase are the same as in E_s (i.e., $C_{2,1}$). The broadcast messages in the first round of reconstruction are $\bar{C}_{1,1}, C_{2,1}$ and $\bar{C}_{3,1}$ respectively. For $i = 2, \ldots, r$, in round i of the reconstruction phase, the parties broadcast the messages $\bar{C}_{1,i}, \bar{C}_{2,i}$ and $\bar{C}_{3,i}$ respectively. As D is honest, due to correctness property of Π, the honest parties need to output s^* at the end of the reconstruction phase.

Next we consider another execution of Π, namely E_s^*.

3. In E_s^*, D is honest and P_3 is faulty. Here D's communication during the first round of sharing is the same as in E_s and the second round broadcast messages of the sharing phase are same as in E_s. However, during the reconstruction phase, P_3 gets corrupted. In the first round of reconstruction, P_3 broadcasts the message $\bar{C}_{3,1}$ (as if he is in execution E_{s^*}), while P_1 and P_2 broadcasts $C_{1,1}$ and $C_{2,1}$ respectively, as in E_s. For $i = 2, \ldots, r$, in round i of the reconstruction phase, the parties broadcast the messages $C'_{1,i}, C'_{2,i}$ and $C'_{3,i}$ respectively. As D is honest, due to correctness property of Π, the honest parties need to output s at the end of the reconstruction phase.

Finally, we consider another execution E of Π, where D $(= P_1)$ is corrupted.

[6] If this is not so, then it implies that B_1, B_2 and B_3 could be generated only for the shares s_1, s_2 and s_3 and a specific randomness, which violates the secrecy condition of protocol Π.

4. In E, during first round of the sharing phase, D gives to parties P_2 and P_3 the shares s_2 and $\bar{s}_3$ respectively. Due to different randomness, the broadcast messages in the second round of the sharing phase are B_1, B_2 and B_3 respectively. During the reconstruction phase, in the first round, P_2 broadcasts $C_{2,1}$ as its view at this point is identical to that in the execution E_s, and P_3 broadcasts $\bar{C}_{3,1}$ as its view is identical to the one in E_{s^*}. Now P_1 can behave in one of the two ways:
   4.1 P_1 behaves as if he is in the reconstruction phase of execution E_{s^*} and broadcasts $\bar{C}_{1,i}$ in i^{th} round of the reconstruction phase for $i = 1, \ldots, r$. Thus the view of P_2 and P_3 at the end of the first round of reconstruction is identical to the view in E_{s^*}. Hence, for $i = 2, \ldots, r$, P_2's and P_3's broadcasts in the i^{th} round of the reconstruction will be the same as in E_{s^*}. Thus at the end of the reconstruction phase, the view of P_2 and P_3 will be same as in E_{s^*} and thus they will reconstruct s^*.
   4.2 P_1 behaves as if he is in the reconstruction phase of execution E_s^* and broadcasts $C_{1,1}$ during first round of the reconstruction phase and $C'_{1,i}$ during i^{th} round of reconstruction phase for $i = 2, \ldots, r$. Now the views of P_2 and P_3 will be the same as in E_s^* at the end of the first round of reconstruction. Using the same arguments as in 4.1 we have that the subsequent rounds of the reconstruction phase will also be the same as in E_s^*, and thus at the end of the reconstruction phase, the parties will output s.

Thus we have shown that a corrupted D can always force during the reconstruction phase the output of the protocol to be one of two secrets, thus violating the weak commitment property. From the above proof, we conclude that there does not exist a 2-round share $(3t, t)$ statistical WSS and hence such a statistical VSS protocol, with any number of rounds in the reconstruction phase. □

5.2 Lower Bound for 1-Round Statistical-VSS

We now derive a non-trivial lower bound on the fault tolerance of any 1-round share statistical VSS (with any number of rounds in reconstruction).

Theorem 7. *1-round share statistical-VSS is possible iff* $((t = 1)$ *and* $(n \geq 4))$*, irrespective of the number of rounds in reconstruction.*

Proof: The impossibility of 1-round share $(3, 1)$ statistical VSS with any number of rounds in reconstruction, follows from Theorem 5. Now we show that for $t \geq 2$ there does not exist any 1-round share (n, t) statistical VSS protocol with $n \geq 4$, irrespective of the number of rounds in the reconstruction phase. We prove the above the statement assuming $t = 2$.

To prove the above claim, we use a hybrid argument. More specifically, we assume that Π is a 1-round share $(n, 2)$ statistical VSS with $n \geq 4$, with any number of rounds in the reconstruction phase. Without loss of generality, let the n parties in Π be denoted by $P_1, \ldots, P_n$ with D being any of these n parties, other than P_1. Before proceeding further, we make the following claim:

Claim. In any execution of Π, the messages broadcast by D and other parties during sharing phase will be independent of the secret. Moreover, private communication between any two honest parties (excluding the ones done from D to the parties) during the sharing phase, will be independent of the messages received from D during the sharing phase.

Proof: From the secrecy property of Π, any message broadcasted by D during the sharing phase should be independent of the secret. Also, since Π has only one round in the sharing phase, the messages exchanged between any two honest parties (excluding the ones given by D to the parties) and the messages broadcasted by the parties during the sharing phase, will be independent of the messages that the parties have received from D during the sharing phase. $\square$

Based on the above claim, we can simply ignore the broadcast done by D and the parties during the sharing phase. We can also ignore all private communication between any two parties (excluding the ones done from D to the parties) during the sharing phase and concentrate only on the messages which are privately communicated by D to the the parties. Thus, without loss of generality, any execution of protocol Π will have the following form:

(Sharing Phase): D, on having a secret Sec, generates messages $Msg_1, \ldots, Msg_n$ and privately communicates Msg_i to party P_i. Since Π is a 1-round share VSS, the sharing phase will take only one round.

(Reconstruction Phase): This may take several rounds. At the end of the reconstruction phase, each party outputs some secret.

Now consider an execution of Π, where an *honest* D, on input secret s, generates $(\alpha_1, \ldots, \alpha_n)$ during sharing phase and privately communicates α_i to P_i. Now from the correctness property of Π, this distribution of messages should output s at the end of the reconstruction phase. We now prove the following claim:

Claim. Any execution of Π, where D (honest or corrupted) generates and distributes $\alpha_1, \ldots, \alpha_{n-1}, \beta_n$ (for any β_n) during the sharing phase, should output the secret s at the end of the reconstruction phase.

Proof: If $\beta_n = \alpha_n$, then the claim is true. Let $\beta_n \neq \alpha_n$, we prove the claim by contradiction. More specifically, let the distribution of messages $\alpha_1, \ldots, \alpha_{n-1}, \beta_n$ outputs secret $s' \neq s$ during the reconstruction phase. Now consider another execution of Π, where D is corrupted and distributes $\alpha_1, \ldots, \alpha_n$ during the sharing phase. During the reconstruction phase, the adversary corrupts P_n and asks him to behave as if P_n has received either α_n or β_n. Accordingly, either s or s' will be reconstructed at the end of the reconstruction phase. This violates the commitment property of Π, which is a contradiction. Hence distribution of the messages $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \beta_n$ (for any β_n) during the sharing phase, should output the secret s at the end of the reconstruction phase. $\square$

Now using similar arguments as in the above claim, we can prove the following lemma:

Lemma 9. *Any execution of Π, where D (honest or corrupted) generates and distributes $\alpha_1, \beta_2, \ldots, \beta_n$ (for any $\beta_2, \ldots, \beta_n$) during the sharing phase, should output the secret s at the end of the reconstruction phase.*

Finally the above lemma clearly shows a violation of the secrecy property of Π because it states that any execution, where D gives message α_1 to P_1 will always output the secret s at the end of the reconstruction phase. So if D is honest and adversary passively corrupts P_1 in such an execution, he will come to know that the shared secret is s, which is a violation of the secrecy property. Theorem 7 now follows from the above discussion. □

Note that the above proof does not hold for WSS due to the fact that WSS requires only weak commitment, this prevents the argument that all sequences of messages sent to the parties need to be reconstructed to the same secret. In fact we can design a 1-round share, 2-round reconstruct $(3t + 1, t)$ statistical WSS protocol (see Appendix A.)

We provide without proof a Statistical VSS, 1-Round Sharing for $n = 4, t = 1$. Proofs are provided in the full version [15].

Sharing Phase

D **selects:** A random polynomial $f(x)$ over $\mathbb{F}$ of degree 1, such that $f(0) = s$. For $i, 2 \leq i \leq 4$ the dealer chooses and sends to P_i the following:

1. A random polynomial $f_i(x)$ over $\mathbb{F}$, $deg(f_i) = 1$ and $f_i(0) = f(i)$.
2. Random non-zero element from $\mathbb{F}$, denoted by α_i.
3. $v_{ji} = f_j(\alpha_i)$ for $2 \leq j \leq 4$.

Reconstruction Phase, 2-rounds: $D(P_1)$ is not allowed to participate

Round 1: Each P_i broadcasts $f_i'(x)$, for $2 \leq i \leq 4$.
Round 2: For $2 \leq i \leq 4$, P_i broadcasts the evaluation point α_i' and the values v_{ji}', for $2 \leq j \leq 4$.
Local Computation (by each party except P_1):

1. party $P_i \in \mathcal{P} \setminus \{P_1\}$ is *confirmed* if there exists a $P_j \in \mathcal{P} \setminus \{P_1, P_i\}$ for which $f_i'(\alpha_j') = v_{ij}'$.
2. If the $f_i'(0)$s corresponding to the set of confirmed parties define a polynomial $f(x)$ of degree one then output $f(0)$ otherwise output NULL.

Open Problems

This paper leaves an interesting open problem: What is the lower bound on the total number of rounds in VSS, i.e. sharing plus reconstruction? This problem is also closely connected to the question of whether we can design a 2-round statistical VSS protocol which satisfies the strong VSS definition. Such a protocol would immediately result in a total of 3-round VSS protocol.

Acknowledgments. We would like to thank the anonymous refrees of CRYPTO 2009 for several helpful suggestions.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: STOC, pp. 1–10 (1988)
2. Chaum, D., Crpeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: FOCS, pp. 11–19 (1988)
3. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In: STOC, pp. 383–395 (1985)
4. Cramer, R., Damgård, I., Dziembowski, S.: On the Complexity of Verifiable Secret Sharing and Multiparty Computation. In: STOC, pp. 325–334 (2000)
5. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient Multiparty Computations Secure Against an Adaptive Adversary. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (1999)
6. Cramer, R., Damgård, I., Maurer, U.M.: General Secure Multi-Party Computation from Any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
7. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly Secure Message Transmission. JACM 40(1), 17–47 (1993)
8. Dwork, C.: Strong Verifiable Secret Sharing. In: van Leeuwen, J., Santoro, N. (eds.) WDAG 1990. LNCS, vol. 486, pp. 213–227. Springer, Heidelberg (1990)
9. Feldman, P.: A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In: FOCS, pp. 427–437 (1987)
10. Fitzi, M., Garay, J., Gollakota, S., Pandu Rangan, C., Srinathan, K.: Round-Optimal and Efficient Verifiable Secret Sharing. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 329–342. Springer, Heidelberg (2006)
11. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In: STOC, pp. 580–589 (2001)
12. Golderich, O., Micali, S., Wigderson, A.: How to Play a Mental Game– a Completeness Theorem for Protocols with Honest Majority. In: STOC 1987, pp. 218–229 (1987)
13. Goldreich, O.: Secure Multiparty Computation (2007), `http://www.wisdom.weizman.ac.il/~oded/pp.html`
14. J. Katz, C. Koo, and R. Kumaresan. Improving the Round Complexity of VSS in Point-to-Point Networks. Cryptology ePrint 2007/358; In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 499–510. Springer, Heidelberg (2008)
15. Patra, A., Choudhary, A., Rabin, T., Pandu Rangan, C.: The Round Complexity of Verifiable Secret Sharing Revisited. Cryptology ePrint 2008/172
16. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
17. Rabin, T.: Robust Sharing of Secrets When the Dealer is Honest or Cheating. J. ACM 41(6), 1089–1109 (1994)
18. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: STOC, pp. 73–85 (1989)
19. Shamir, A.: How to Share a Secret. Comm. of the ACM 22(11), 612–613 (1979)
20. Tompa, M., Woll, H.: How to Share a Secret with Cheaters. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 261–265. Springer, Heidelberg (1986)

A 1-Round Statistical WSS

For (partial) completeness we present the 1-Round Statistical WSS for $n = 3t+1$. Proofs appear in the full version [15].

Sharing Phase

Local computations: D picks:

1. a random polynomial $G(x)$ over $\mathbb{F}$ of degree t, such that $G(0) = s$.
2. n random polynomials $g_1(x), g_2(x), \ldots, g_n(x)$ over $\mathbb{F}$, each of degree t, such that for $1 \leq i \leq n$, $g_i(0) = G(i)$.
3. n random non-zero distinct elements from $\mathbb{F}$, denoted by $\alpha_1, \alpha_2, \ldots, \alpha_n$.

D's communication: D sends to party P_i:

- the polynomial $g_i(x)$,
- the random value α_i and the n tuple $[v_{1i}\ v_{2i}\ \ldots\ v_{ni}]$ where for $1 \leq j \leq n$, $v_{ji} = g_j(\alpha_i)$.

Reconstruction Phase, 2-rounds

Round 1: Each $P_i \in \mathcal{P}$ broadcasts the polynomial $g'_i(x)$.
Round 2: Each $P_i \in \mathcal{P}$ broadcasts the value α'_i and the n tuple $[v'_{1i}\ v'_{2i}\ \ldots\ v'_{ni}]$.

1. Party P_i *accepts* party P_j if $v'_{ji} = g'_j(\alpha'_i)$.

2. Party P_i is called *affirmed* if it is *accepted* by at least $2t + 1$ parties (possibly including itself) where as P_i is called *semi-affirmed* if it is accepted by at least $t + 1$ and by at most $2t$ parties (possibly including itself).

Local Computation: (For all parties)

1. IF the number of *affirmed* parties is less than $2t + 1$ OR the number of *semi-affirmed* parties is more than zero, then output $NULL$.
2. ELSE let $CORE$ be the set of all *affirmed* parties. Consider $g'_i(0)$'s of all the parties in $CORE$ and check whether they interpolate a unique t degree polynomial, say $G'(x)$. IF yes, then output $s' = G'(0)$, ELSE output $NULL$.

Somewhat Non-committing Encryption and Efficient Adaptively Secure Oblivious Transfer[⋆]

Juan A. Garay[1], Daniel Wichs[2], and Hong-Sheng Zhou[3,⋆⋆]

[1] AT&T Labs – Research
garay@research.att.com
[2] New York University
wichs@cs.nyu.edu
[3] University of Connecticut
hszhou@cse.uconn.edu

Abstract. Designing efficient cryptographic protocols tolerating adaptive adversaries, who are able to corrupt parties on the fly as the computation proceeds, has been an elusive task. In this paper we make progress in this area. First, we introduce a new notion called *semi-adaptive* security which is slightly stronger than static security but *significantly weaker than fully adaptive security.* The main difference between adaptive and semi-adaptive security is that semi-adaptive security allows for the case where one party starts out corrupted and the other party becomes corrupted later on, but *not* the case where both parties start out honest and become corrupted later on. As such, semi-adaptive security is much easier to achieve than fully adaptive security. We then give a simple, generic protocol compiler which transforms any semi-adaptively secure protocol into a fully adaptively secure one. The compilation effectively decomposes the problem of adaptive security into two (simpler) problems which can be tackled separately: the problem of semi-adaptive security and the problem of realizing a weaker variant of secure channels.

We solve the latter problem by means of a new primitive that we call *somewhat non-committing encryption* resulting in significant efficiency improvements over the standard method for realizing secure channels using (fully) non-committing encryption. Somewhat non-committing encryption has two parameters: an equivocality parameter ℓ (measuring the number of ways that a ciphertext can be "opened") and the message sizes k. Our implementation is very efficient for small values ℓ, *even* when k is large. This translates into a very efficient compilation of semi-adaptively secure protocols for tasks with small input/output domains (such as bit-OT) into fully adaptively secure protocols.

Indeed, we showcase our methodology by applying it to the recent Oblivious Transfer protocol by Peikert *et al.* [Crypto 2008], which is only secure against static corruptions, to obtain the first efficient, adaptively secure and composable OT protocol. In particular, to transfer an n-bit message, we use a constant number of rounds and $O(n)$ public key operations.

⋆ Partial work was carried out when the first author was at Bell Labs and the second and the third authors were visiting Bell Labs.
⋆⋆ Research supported by NSF grants 0447808 and 0831306.

1 Introduction

When defining the security of cryptographic protocols, we generally strive to capture as wide a variety of adversarial attacks as possible. The most popular method of doing so is the *simulation paradigm* [17] where the security of a real-world protocol is compared to that of an ideal-world (perfectly secure) implementation of the same task. Within the simulation paradigm there are several flavors. Firstly, basic simulation only guarantees security for single copy of a protocol executing in isolation. The *Universal Composability* (UC) framework [3,4] extends the simulation paradigm and defines security for protocols executed in arbitrary environments, where executions may be concurrent and even maliciously interleaved. Secondly, we generally distinguish between *static* and *adaptive* security. Static security protects against an adversary who controls some fixed set of corrupted parties throughout the computation. Adaptive security, on the other hand, defends against an adversary who can corrupt parties adaptively at any point during the course of the protocol execution For adaptive security, we also make a distinction between the *erasure model*, where honest parties are trusted to securely erase data as mandated by the protocol, and the *non-erasure model*, where no such assumptions are made. Traditionally, the design of protocols in the non-erasure model is viewed as significantly more difficult. For example, and in contrast to the erasure model, we do not have general constant-round protocols for many tasks, and even many simple tasks (e.g., encryption) seem to be less efficient in rounds and computation. Nevertheless, although solutions in the erasure model may in some scenarios be acceptable, it is both of fundamental interest and practical value to achieve the stronger security notion whenever possible; this is the subject of this work.

The seminal result of [8] shows that it is theoretically possible to design an adaptively secure and universally composable protocol for a large class of natural tasks, assuming the presence of some trusted setup such as a randomly selected common reference string (CRS). Unfortunately, the final protocol of [8] should be viewed as a *purely theoretical* construction, ad its reliance on expensive Cook-Levin reductions precludes a practical implementation. Alternative efficient approaches to two-party and multi-party computation received a lot of attention in the recent works of [13,16,20,21,22,24,25]. However, all of these results sacrifice some aspect of security to get efficiency – e.g., they only provide (stand-alone or UC) static security, or UC adaptive security but only for honest majority, or UC adaptive security in the erasure model, etc. The recent work of [20] *can* provide UC adaptive security for all (well-formed) tasks in constant rounds assuming the adversary corrupts all but one of the participants, but only given an efficient adaptively secure Oblivious Transfer (OT) protocol. However, as we will discuss, no such protocols were known. Lastly, we mention the work of [5], which gives a generic compiler from static to adaptive security using secure channels. Unfortunately, this compiler does not provide full adaptive security (does not allow for post-execution corruptions) and, as was noted in [24], crucially relies on rewinding and hence cannot be used in the UC framework.

Indeed, thus far *no* efficient protocols for general multi-party computation, or even for many specific two-party function evaluation tasks, achieve adaptive security. This is not surprising given the difficulty of realizing adaptive security for even the most fundamental task in cryptography: *secure communication.* As was observed in [6], standard security notions for encryption do not suffice. Adaptively secure communication schemes, also called *non-committing encryption* schemes, were introduced and constructed in [6] and studied further in [1,11], but these protocols are fairly complicated and inefficient for large messages.

It turns out that many useful two-party tasks (e.g., Oblivious Transfer, OR, XOR, AND, Millionaires' problem, etc.) are *strictly harder* to achieve than secure communication, the reason being that these tasks allow two honest parties to communicate by using the corresponding ideal functionality. For example, using OT, an honest sender can transfer a message to a receiver by setting it as *both* of his input values. Therefore, an adaptively secure OT protocol for the transfer of k bit messages can be used as a non-committing encryption of a k bit message and so all of the difficulty and inefficiency of non-committing encryption must **also** appear in protocols for tasks such as OT. Further, unlike secure communication, many tasks also require security against the active and malicious behavior of the *participants.* This might lead us to believe that the two difficulties will be compounded making efficient adaptively secure implementations of such tasks infeasible or too complicated to contemplate.

Taking Oblivious Transfer as an example, this indeed seems to be the case. The recent work of [26], proves a (black-box) separation between enhanced trapdoor permutations (which allow for static OT) and adaptively secure OT, showing that the latter is indeed "more complex" in a theoretical sense. This complexity is reflected in practice as well. We are aware of only two examples (albeit inefficient) of adaptively secure OT protocols, from [2] and [8]. Both of these works first construct an OT protocol for the honest-but-curious setting and then compile it into a fully secure protocol using generic and inefficient zero knowledge proofs. In both constructions, the underlying honest-but-curious OT protocols rely on ideas from non-committing encryption[1] and hence inherit its complexity. Since the full constructions require us to run zero knowledge proofs *on top of* the complex underlying honest-but-curious protocol, there is little hope of making them efficient by only using proofs for simple relations. This is in contrast to static security (and adaptive security in the erasure model) for which we *have* recently seen efficient constructions of OT protocols. For example, [14,16,21] construct OT protocols by only using simple and efficient zero-knowledge proofs. Interestingly, Ishai *et al.* [19] give the first OT protocol constructions against malicious corruptions *without* using zero knowledge proofs; this result was later strengthened in [18]. Two very recent and efficient concrete protocols not using zero-knowledge proofs are given in [23,28]. The protocol of [28] is particularly exciting since it is a UC-secure protocol in the CRS model which runs in two rounds and uses a constant number of

[1] The protocol of [2] implicitly uses the plug-and-play approach from [1], while the protocol of [8] uses non-committing encryption in a generic way.

public key operations. Achieving adaptive security based on these protocols has, however, remained as an open problem.

Our contributions. In this work we construct the first efficient (constant round, $O(n)$ public-key operations for the transfer of an n-bit message) adaptively secure Oblivious Transfer protocol in the non-erasure model. Along the way we develop several techniques of independent interest which are applicable to adaptive security in general.

First, we introduce a new notion called *semi-adaptive* security which is slightly stronger than static security but significantly weaker than fully adaptive security. At a high level, semi-adaptive security allows for the case where one party starts out corrupted and the other party becomes corrupted later on, but *not* the case where both parties start out honest and become corrupted later on. In particular, a semi-adaptively secure protocol for a task like OT, *does not* yield a non-committing encryption scheme and hence does not (necessarily) inherit its difficulty. We then give a generic compiler which transforms any semi-adaptively secure protocol into a (fully) adaptively secure protocol. The compiler is fairly simple: we take the original protocol and execute it over a secure communication channel (i.e., all communication from one party to another is sent over a secure channel). The compilation effectively decomposes the problem of adaptive security into two (simpler) problems which can be tackled separately: the problem of semi-adaptive security and the problem of realizing secure channels.

Unfortunately, we saw that the construction of secure-channels is a difficult problem and existing solutions are not very efficient. Also, as we already mentioned, we cannot completely bypass this problem since adaptive security for many tasks *implies* secure channels. However, for the sake of efficiency, we would like to limit the use of secure channels (and hence the use of non-committing encryption) to a minimum. For example, we know that an OT protocol for one-bit messages implies a non-committing encryption of a one-bit message. However, to get adaptive security for a bit-OT protocol, our compiler, as described above, would use non-committing encryption to encrypt the *entire* protocol transcript, and hence much more than one bit!

We fix this discrepancy by introducing a new notion called *somewhat non-committing encryption.* Somewhat non-committing encryption has two parameters: the equivocality ℓ (measuring just how *non-committing* the scheme is) and the message size k. We first observe that somewhat non-committing encryption is efficient for small values of the equivocality parameter ℓ, *even* when k is large (i.e., when we encrypt long messages). Secondly, we observe that our compiler can use somewhat non-committing encryption where the equivocality ℓ is proportional to the size of the input and output domains of the functionality. As a result, we obtain a very efficient compiler transforming any semi-adaptively secure protocol for a task with small input/output domains (such as bit-OT) into a fully adaptively secure protocol. We also show that this methodology can, in special cases, be applied to tasks with larger domain sizes such as string-OT with long strings.

We apply our methodology to the OT protocol of Peikert *et al.* [28], resulting in the first efficient and adaptively secure OT protocols. Peikert *et al.* actually present a general framework for constructing static OT, and instantiate this framework using the Quadratic Residuocity (QR), Decisional Diffie-Hellman (DDH), and Lattice-based assumptions. In this work, we concentrate on the QR and DDH based schemes. We show that relatively small modifications suffice to make these schemes semi-adaptively secure. We then employ our compiler, using somewhat non-committing encryption, to convert them into (fully) adaptively UC-secure OT protocols. As we mentioned previously, the work of [20] shows how to efficiently realize all (well-formed) m-party functionalities with adaptive security assuming up to $m-1$ corruptions, given and adaptively secure OT protocol. Therefore, by plugging in our OT protocol construction, we get improved efficiency in generically compiled protocols for all tasks as above.

Concurrent and independent work. Following the line of work of [19,18], the recent result by Choi *et al.* [9] gives a generic black-box compiler from semi-honest adaptively secure OT to fully malicious adaptively secure OT, using cut-and-choose techniques. Although the end result of our work is the same (adaptively secure OT), the two works take very different approaches which complement each other well: the compiler of [9] transforms semi-honest + adaptive security into malicious + adaptive security in the special case of OT, while our compiler is a general transformation from malicious + semi-adaptive security to malicious + adaptive security. The two starting notions of security (semi-honest + adaptive vs. malicious + semi-adaptive) are incomparable and thus both compilers are useful in different scenarios. In particular, our compiler can be used in conjunction with the OT protocol of [28] and results in an extremely efficient adaptively secure OT protocol using a constant number of rounds and $O(n)$ public-key operations to transfer an n-bit string.[2] In contrast, the compiler of [9] shows how to base adaptively secure OT on a simulatable cryptosystem in a black-box way, but at the expense of running $\Omega(\lambda^2)$ copies of the underlying semi-honest OT protocol, where λ is the security parameter, and thus requiring $\Omega(\lambda^2 n)$ operations for n-bit OT. Therefore our OT protocol can be significantly more efficient.

Due to space limitations, proofs, together with background material, full description of our enhanced version of the QR dual-mode cryptosystem, efficiency considerations, and our DDH version of adaptively secure bit- and string-OT, can be found in the full version of the paper [15].

2 Somewhat Non-committing Encryption

2.1 Adaptive Security in Two-Party Protocols

What are some of the challenges in achieving adaptive security for a two-party protocol? Let's assume that a protocol π between two parties P_0, P_1 realizes a

[2] Technically, if one thinks of n as a function of λ, we require $O(\max(\lambda, n))$ operations.

task $\mathcal{F}$ with respect to static adversaries. That means that there is a static simulator which can simulate the three basic cases: both parties are honest throughout the protocol, exactly one party is corrupted throughout the protocol or both parties are corrupted throughout the protocol. To handle adaptive adversaries, we require two more capabilities from our simulator: the ability to simulate a *first corruption* (i.e., the case that both parties start out honest and then one of them becomes corrupted) and simulating the *second corruption* (i.e., one party is already corrupted and the other party becomes corrupted as well).

Simulating the first corruption is often the harder of the two cases. The simulator must produce the internal state for the corrupted party in a manner that is consistent with the protocol transcript so far and with the actual inputs of that party (of which the simulator had no prior knowledge). Moreover, the simulator needs to have all the necessary trapdoors to continue the simulation *while* only one party is corrupted. Achieving both of these requirements at once is highly non-trivial and this is one of the reasons why efficient protocols for adaptively secure two-party computation have remained elusive.

Interestingly, simulating the first corruption becomes much easier if the protocol π employs secure channels for all communication between parties. At a high level, the simulator does not have to do any work while both parties are honest, since the real-world adversary does not see any relevant information during this time! When the first party *becomes* corrupted, we can just run a static simulation for the scenario in which this party *was* corrupted from the beginning but acting honestly and using its input. Then, we can "lie" and pretend that this communication (generated *ex post facto*) actually *took place* over the secure channel when both parties were honest. The lying is performed by setting the internal state of the corrupted party accordingly. Since our lie corresponds to the simulation of a statically corrupted party (which happens to act honestly), all of the trapdoors are in place to handle future mischievous behavior by that (freshly corrupted) party. The only problem left is in handling the second corruption – but this is significantly easier! To formalize this, we will define a notion of *semi-adaptive security* where the simulator needs to be able to simulate static corruptions as well as the case where one party starts out corrupted and the other party becomes corrupted later on (but *not* the case where *both* parties start out honest and may become corrupted later). The formal notion (with some additional restrictions imposed on the simulator) appears in Section 2.4.

Informally, we have argued that if two-party protocol is semi-adaptively secure, then the protocol is also fully adaptively secure if all communication between the parties is sent over an idealized secure channel. Unfortunately, idealized secure channels are hard to achieve physically and implementing such channels cryptographically in the real world requires the inefficient use of *non-committing encryption* [6] to encrypt the entire protocol transcript. Luckily, it turns out that we often do not need to employ fully non-committing encryption to make the above transformation hold. Indeed, we define a weaker primitive called *somewhat non-committing encryption* and show that this primitive can be implemented with significantly greater efficiency than (fully) non-committing encryption, and that it is often *good enough* to transform a semi-adaptively secure protocol into a fully adaptively secure protocol when the sizes of the input/output domains are small.

2.2 Defining *Somewhat* Non-committing Encryption

First recall the notion of non-committing encryption from [6], which is a protocol used to realize secure channels in the presence of an *adaptive* adversary. In particular, this means that a simulator can produce "fake" ciphertexts and later explain them as encryptions of *any possible* given message. Several non-committing encryption schemes have appeared in literature [6,1,11], but the main disadvantage of such schemes is the computational cost. All of the schemes are interactive (which was shown to be necessary in [27]) and the most efficient schemes require $\Omega(1)$ public-key operations *per bit* of plaintext.

We notice that it is often unnecessary to require that the simulator can explain a ciphertext as the encryption of *any* later-specified plaintext. Instead, we define a new primitive, which we call *somewhat non-committing* encryption, where the simulator is given a set of ℓ messages during the generation of the fake ciphertext and must later be able to plausibly explain the ciphertext as the encryption of *any one of those ℓ messages*. In a sense, we distinguish between two parameters: the plaintext size (in bits) k and the equivocality ℓ (the number of messages that the simulator can plausibly explain). For fully non-committing encryption, the equivocality and the message size are related by $\ell = 2^k$. Somewhat non-committing encryption, on the other hand, is useful in accommodating the case where the equivocality ℓ is very small, but the message size k is large.

Functionality $\mathcal{F}_{\text{SC}}^{\mathcal{N}}$

The ideal functionality $\mathcal{F}_{\text{SC}}^{\mathcal{N}}$ interacts with an initiator I and a receiver R. It consists of a channel-setup phase, after which the two parties can send arbitrarily many messages from one to another. The functionality is parameterized by a non-information oracle $\mathcal{N}$.

Channel setup: Upon receiving $(\texttt{ChSetup}, sid, I)$ from party I, initialize the machine $\mathcal{N}$ and record the tuple $(sid, \mathcal{N})$. Pass the message $(\texttt{ChSetup}, I)$ to R. In addition, pass this message to $\mathcal{N}$ and forward its output to the adversary $\mathcal{S}$.

Message transfer: Upon receiving $(\texttt{Send}, sid, P, m)$ from party P where $P \in \{I, R\}$, find a tuple $(sid, \mathcal{N})$ and, if none exists, ignore the message. Otherwise, send the message $(\texttt{Send}, sid, P, m)$ to the *other* party $\overline{P} = \{I, R\} - \{P\}$. In addition, invoke $\mathcal{N}$ with $(\texttt{Send}, sid, P, m)$ and forward its output to $\mathcal{S}$.

Corruption: Upon receiving a message $(\texttt{Corrupt}, sid, P)$ from the adversary, send $(\texttt{Corrupt}, sid, P)$ to $\mathcal{N}$ and forward its output to $\mathcal{S}$. After the first corruption, stop the execution of $\mathcal{N}$ and give $\mathcal{S}$ complete control over the functionality (i.e., $\mathcal{S}$ learns all inputs and can specify any outputs).

Fig. 1. The parameterized secure-channel ideal functionality, $\mathcal{F}_{\text{SC}}^{\mathcal{N}}$

It is challenging to define an ideal-functionality for *somewhat* non-committing encryption, since the ideal world captures a notion of security which is too strong. Here, we take the approach of [7] where ideal-world functionalities are weakened by the inclusion of a *non-information oracle* which is a PPT TM that captures the information leaked to the adversary in the ideal world. The ideal functionality for secure channels, given in Figure 1, is parameterized using a non-information oracle $\mathcal{N}$ which gets the values of the exchanged messages m and outputs some side information to the adversary $\mathcal{S}$. The security of the secure channel functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}}$ depends on the security properties required for the machine $\mathcal{N}$ and thus we can capture several meaningful notions. Let us first start with the most secure option which captures (fully) non-committing encryption.

Definition 1. *Let $\mathcal{N}^{\mathrm{full}}$ be the oracle, which, on input $(\mathtt{Send}, sid, P, m)$, produces the output $(\mathtt{Send}, sid, P, |m|)$ and, on any inputs corresponding to the* $\mathtt{ChSetup}, \mathtt{Corrupt}$ *commands, produces no output. We call the functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}^{\mathrm{full}}}$, or just $\mathcal{F}_{\mathrm{SC}}$ for brevity, a* (fully) non-committing secure channel. *A real-world protocol which realizes $\mathcal{F}_{\mathrm{SC}}$ is called a* non-committing encryption scheme (NCE)*.*

Above, the oracle $\mathcal{N}$ never reveals anything about messages m exchanged by two honest parties, even if (both of the) parties later get corrupted. Hence the functionality is fully *non-committing.* To define *somewhat* non-committing encryption we first give the following definitions of non-information oracles.

Definition 2. *A machine $\mathcal{R}$ is called a* message-ignoring *oracle if, on any input $(\mathtt{Send}, sid, P, m)$, it ignores the value m and processes only the input $(\mathtt{Send}, sid, P, |m|)$. A machine $\mathcal{M}$ called a* message-processing *oracle if it has no such restrictions. We call a pair of machines $(\mathcal{M}, \mathcal{R})$* well-matched *if no PPT distinguisher $\mathcal{D}$ (with oracle access to either $\mathcal{M}$ or $\mathcal{R}$) can distinguish the message-processing oracle $\mathcal{M}$ from the message-ignoring oracle $\mathcal{R}$.*

We are now ready to define the non-information oracle used by a somewhat non-committing secure channel ideal functionality.

Definition 3. *Let $(\mathcal{M}, \mathcal{R})$ be a well-matched pair which consists of a message-processing and a message-ignoring oracle respectively. Let $\mathcal{N}^{\ell}$ be a (stateful) oracle with the following structure.*

- *Upon initialization, $\mathcal{N}^{\ell}$ chooses a uniformly random index $i \overset{\$}{\leftarrow} \{1, \ldots, \ell\}$. In addition it initializes a tuple of ℓ* independent *TMs: $\langle \mathcal{N}_1, \ldots, \mathcal{N}_\ell \rangle$ where $\mathcal{N}_i = \mathcal{M}$ and, for $j \neq i$, the machines $\mathcal{N}_j$ are independent copies of the message-ignoring oracle $\mathcal{R}$.*
- *Whenever $\mathcal{N}^{\ell}$ receives inputs of the form $(\mathtt{ChSetup}, sid, P)$ or $(\mathtt{Send}, sid, P, m)$, it passes the input to each machine $\mathcal{N}_i$ receiving an output y_i. It then outputs the vector $(y_1, \ldots, y_\ell)$.*
- *Upon receiving an input $(\mathtt{Corrupt}, sid, P)$, the oracle reveals the internal state of the message-processing oracle $\mathcal{N}_i$ only.*

For any such oracle $\mathcal{N}^\ell$, we call the functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}^\ell}$ an ℓ-equivocal non-committing secure channel. *For brevity, we will also use the notation $\mathcal{F}_{\mathrm{SC}}^{\ell}$ to denote $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}^\ell}$ for some such oracle $\mathcal{N}^\ell$. Lastly, a real world protocol which realizes $\mathcal{F}_{\mathrm{SC}}^{\ell}$ is called an* ℓ-equivocal non-committing encryption scheme (ℓ-NCE).

As before, no information about messages m is revealed during the "send" stage. However, the internal state of the message-processing oracle $\mathcal{N}_i$, which is revealed upon corruption, might be "committing." Nevertheless, a simulator can simulate the communication between two honest parties over a secure channel, as modeled by $\mathcal{F}_{\mathrm{SC}}^{\ell}$, in a way that allows him to later explain this communication as any one of ℓ possibilities. In particular, the simulator creates ℓ message-processing oracles and, for every `Send` command, the simulator chooses ℓ distinct messages $m_1, \ldots, m_\ell$ that he passes to the oracles $\mathcal{M}_1, \ldots, \mathcal{M}_\ell$ respectively. Since message-processing and message-ignoring oracles are indistinguishable, this looks indistinguishable from the side information produced by $\mathcal{F}_{\mathrm{SC}}^{\ell}$. Later, when a corruption occurs, the simulator can convincingly explain the entire transcript of communication to any one of the ℓ possible options, by providing the internal state of the appropriate message-processing oracle $\mathcal{M}_i$.

2.3 The ℓ-NCE Scheme Construction

The construction of ℓ-NCE is based on a *simulatable public-key system* [11], wherein it is possible to generate public keys obliviously, without knowing the corresponding secret key, and to explain an honestly (non-obliviously) generated public key as one which was obliviously generated. In a similar way, there should be a method for obliviously generating ciphertexts (without knowing any plaintext) and to explain honestly generated (non-oblivious) ciphertexts as obliviously generated ones. Refer to the full version for review of the syntax and security properties of such a scheme. Our ℓ-NCE protocol construction, shown in Figure 2, uses a fully non-committing secure channel, but only to send a *very short* message during the setup phase. In addition, it uses a simulatable public-key system and a symmetric key encryption scheme where ciphertexts are indistinguishable from uniformly random values (the latter can be constructed from any one way function). For very long communications and small ℓ, our ℓ-NCE scheme is significantly more efficient than (full) NCE.

Theorem 1. *The protocol in Figure 2 is an ℓ-NCE scheme. Specifically, it UC-realizes functionality $\mathcal{F}_{\mathrm{SC}}^{\ell}$ in the presence of an active and adaptive adversary.*

The main efficiency consideration is the use of fully non-committing encryption of the index i (which is small). We show in the full version that our scheme uses a total of expected $\mathcal{O}(\log \ell)$ public-key operations, expected $\mathcal{O}(\ell\lambda)$ communication and expected constant rounds of interaction for the channel setup phase, where λ is the security parameter. Alternatively, if one would like to set up $n = \Omega(\lambda)$ channels in parallel, this can be done in strict $\mathcal{O}(n \log \ell)$ public-key operations, strict $\mathcal{O}(n\ell\lambda)$ communication and strict constant number of rounds of interaction. After channel-setup, encryption is non-interactive and requires only

Let $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a *simulatable public-key system* and $\widetilde{\mathsf{KG}}$, $\widetilde{\mathsf{Enc}}$ be the corresponding *oblivious key generator* and *oblivious ciphertext generator* algorithms. Further, let $(\mathsf{KG}^{\mathsf{sym}}, \mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ be a symmetric-key encryption scheme in which ciphertexts are indistinguishable from uniformly random values of the same length.

Channel Setup. An initiator I sets up a channel with a receiver R as follows:

1. The initiator I sends a random index $i \in \{1, \ldots, \ell\}$ to R over a fully non-committing secure channel.
2. The initiator I generates ℓ public keys. For $j \in \{1, \ldots, \ell\} \setminus \{i\}$, the keys $pk_j \leftarrow \widetilde{\mathsf{KG}}()$ are sampled obliviously, while $(pk_i, sk_i) \leftarrow \mathsf{KG}()$ is sampled correctly. The keys $pk_1, \ldots, pk_\ell$ are sent to R while I stores sk_i.
3. The receiver R chooses a random key $K \leftarrow \mathsf{KG}^{\mathsf{sym}}$ and computes $C_i = \mathsf{Enc}_{pk_i}(K)$ correctly. In addition, R samples $C_j \leftarrow \widetilde{\mathsf{Enc}}_{pk_j}()$ obliviously for $j \in \{1, \ldots, \ell\} \setminus \{i\}$ and sends the ciphertexts $C_1, \ldots, C_\ell$ to I.
4. The initiator I decrypts the key $K \leftarrow \mathsf{Dec}_{sk_i}(C_i)$. Both parties store (K, i).

Encryption. An initiator I encrypts a message m to a receiver R as follows:

1. The initiator I computes $E_i \leftarrow \mathsf{Enc}^{\mathsf{sym}}_K(m)$ and chooses E_j for $j \in \{1, \ldots, \ell\} \setminus \{i\}$ as uniformly random and independent values of length $|E_i|$. The tuple $(E_1, \ldots, E_\ell)$ is sent to R.
2. The receiver R ignores all values other than E_i. It computes $m \leftarrow \mathsf{Dec}^{\mathsf{sym}}_K(E_i)$.

Fig. 2. The ℓ-NCE protocol

symmetric-key operations. However, the encryption of a k bit message requires $\mathcal{O}(\ell k)$ bits of communication.

2.4 The Adaptive Security Protocol Compiler for Two-Party SFE

As an application of ℓ-NCE, we now give a general theorem showing that a protocol with semi-adaptive security can be compiled into a protocol with (full) adaptive security when all of the communication is encrypted using ℓ-NCE for some appropriate ℓ. However, we must first give a formal definition of semi-adaptive security.

Definition 4. *An adversarial strategy is* <u>*second-corruption adaptive*</u> *if* either *at least one of the parties is corrupted prior to protocol execution* or *no party is ever corrupted. In the former case, the other party can be adaptively corrupted at any point during or after protocol execution.*

Intuitively, we would like to say that a protocol is semi-adaptively secure if it is secure with respect to second-corruption adaptive strategies. Unfortunately, there are two subtleties that we must consider. Firstly, we know that most tasks cannot be realized in the Universal Composability framework without the use of *trusted setup*. However, the use of trusted setup complicates our transformation. The point of using (somewhat) non-committing encryption is that the simulator can lie about *anything that occurs while both parties are honest.* However, we often rely on trusted setup in which some information is given to the adversary even when both parties are honest. For example, the usual modeling of a common reference string specifies that this string is made public and given to the adversary even when *none* of the participants in the protocol are corrupted. In this case the simulator is committed to such setup even if the parties communicate over secure channels. Therefore we require that, when trusted setup is used, the semi-adaptive simulator simulates this setup independently of which party is corrupted. We call this property *setup-adaptive simulation.*

The second subtlety comes from the following type of problem. As we outlined in our informal discussion, we wish to run the semi-adaptive simulator once the first party gets corrupted and then "lie" that the simulated conversation took place over the secure channel. However, when the first party gets corrupted after the protocol execution, then the ideal functionality has already computed the outputs using the honest inputs and will therefore not accept anymore inputs from the semi-adaptive simulator. Recall that we run the semi-adaptive simulator with respect to an adversary $\mathcal{A}$ which follows the protocol execution using the corrupted party's honest input x. If the semi-adaptive simulator extracts the same input x as the one used by $\mathcal{A}$, then we also know the corresponding output and can give it to the semi-adaptive simulator on behalf of the ideal functionality. Therefore it is crucial that the semi-adaptive simulator can only submit the actual input x. We call this property *input-preserving.* Putting Definition 4 and the above notions together, we are finally ready to define semi-adaptive security.

Definition 5. *We say that a protocol π semi-adaptively realizes the ideal functionality $\mathcal{F}$ if there exists a setup-adaptive and input-preserving PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$ and environment $\mathcal{Z}$ which follow a second-corruption adaptive adversarial strategy, we have $REAL_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.*

Lastly, we define the notion of a *well-structured* protocol. Since even non-committing encryption *commits* the simulator to the lengths of the exchanged messages, the number of such messages, and the identities of the sender and receiver of each message, we require that this information is fixed and always the same any given execution of a protocol. Almost all known constructed protocols for cryptographic tasks are well-structured and any protocol can be easily converted into a well-structured protocol.

First we look at the simple compiler using idealized secure channels.

Theorem 2. *Let $\mathcal{F}_{\mathrm{SFE}}^{f}$ be the two-party ideal functionality which computes some function f as defined in Figure 3. Assume that a well-structured two-party protocol π for $\mathcal{F}_{\mathrm{SFE}}^{f}$ is semi-adaptively secure. Let π' be the protocol in which the parties run π but only communicate with each other using non-committing secure channels as modeled by $\mathcal{F}_{\mathrm{SC}}$. Then π' is (fully) adaptively secure.*

As we already mentioned, this compiler is usually not very efficient because of its excessive use of secure channels and hence NCE. Recall that secure channels are employed so that, when both parties are honest, the adversary does not see

Functionality $\mathcal{F}^f_{\text{SFE}}$

The functionality $\mathcal{F}^f_{\text{SFE}}$ interacts with an *initiator* I and a *responder* R.

Input: Upon receiving the input value $(\texttt{Input}_I, sid, x_I)$ from the initiator I, record the value $\langle I, x_I\rangle$ and send the message $(\texttt{Input}_I, sid)$ to the adversary $\mathcal{S}$. Ignore future $(\texttt{Input}_I, \ldots)$ inputs. Similarly, upon receiving the input value $(\texttt{Input}_R, sid, x_R)$ from the responser R, record the value $\langle R, x_R\rangle$ and send the message $(\texttt{Input}_R, sid)$ to the adversary $\mathcal{S}$. Ignore future $(\texttt{Input}_R, \ldots)$ inputs.

Output: Upon receiving the message $(\texttt{Output}_I, sid)$ from the adversary $\mathcal{S}$, if either $\langle I, x_I\rangle$ or $\langle R, x_R\rangle$ is not recorded, ignore the message. Else if $\langle y_I, y_R\rangle$ is not recorded, then compute $(y_I, y_R) \leftarrow f(x_I, x_R)$ and record $\langle y_I, y_R\rangle$; send the output value $(\texttt{Output}_I, sid, y_I)$ to I. Ignore future $(\texttt{Output}_I, \ldots)$ messages from the adversary. Similarly, upon receipt of $(\texttt{Output}_R, sid)$ from the adversary, send the output value $(\texttt{Output}_R, sid, y_R)$ to R. Ignore future $(\texttt{Output}_R, \ldots)$ messages from the adversary.

Fig. 3. Two-party secure evaluation functionality for $f : X_I \times X_R \to Y_I \times Y_R$

any useful information and so this case is easy to simulate. Then, when the first party gets corrupted, our simulator simply makes up the transcript of the communication that should have taken place *ex post facto.* This transcript is generated based on which party got corrupted, what its inputs were and what its outputs were. However, we notice that for many simple protocols there are not too many choices for this information. The simulator must simply be able to credibly lie that the communication which took place over the secure channel corresponds to any one of these possible choices. Using this intuition, we show that a more efficient compiler using ℓ-NCE (for some small ℓ) suffices.

Theorem 3. *Let $\mathcal{F}^f_{\text{SFE}}$ be the two-party ideal functionality computing some function $f : X_I \times X_R \to Y_I \times Y_R$, as defined in Figure 3. Assume that a well-structured two-party protocol π for $\mathcal{F}^f_{\text{SFE}}$ is semi-adaptively secure. Let π' be the protocol in which the parties run π but only communicate with each other using ℓ-equivocal secure channels as modeled by $\mathcal{F}^\ell_{\text{SC}}$ where $\ell = |X_I||Y_I| + |X_R||Y_R|$. Then π' is (fully) adaptively secure.*

3 Efficient and Adaptively Secure Oblivious Transfer

We now apply our compiler of Theorem 3 to the concrete problem of bit- and string-OT, resulting in the first efficient protocols for this task. Refer to [4,15] for the specification of an ideal functionality for OT.

3.1 The PVW Oblivious Transfer Protocol

In [28], Peikert *et al.* construct an efficient OT protocol in the CRS model with UC security against a malicious but static adversary. They do so by introducing a new primitive called a *dual-mode cryptosystem*, which almost immediately yields an OT protocol in the CRS model, and give constructions of this primitive under the DDH, QR and lattice hardness assumptions. We first present a brief review of dual-mode encryption as in [28], and then will define a modified version of this primitive which will allow us to get adaptive security.

A dual-mode cryptosystem is initialized with *system parameters* which are generated by a trusted third party. For any choice of system parameters, the cryptosystem has two types of public/private key pairs: *left* key pairs and *right* key pairs. The key-generation algorithm can sample either type of key pair and the user specifies which type is desired. Similarly, the encryption algorithm can generate a *left* encryption or a *right* encryption of a message. When the key pair type matches the encryption type (i.e. a left encryption of a message under a left public key) then the decryption algorithm (which uses the matching secret key) correctly recovers the message.

As shown in [28], a dual-mode cryptosystem can be used to get an OT protocol, as follows. The receiver chooses to generate a left or right key depending on his input bit σ, and the sender uses left-encryption ($b = 0$) for the left message x_0 and right-encryption for the right message. The receiver then uses the secret key to correctly decrypt the chosen message.

Security against malicious (static) adversaries in the UC model relies on the two different modes for generating the system parameters: *messy* mode and *decryption* mode. In *messy mode*, the system parameters are generated together with a *messy trapdoor.* Using this trapdoor, any public key (even one which is maliciously generated) can be easily labeled a left key or a right key. Moreover, in messy mode, when the encryption type does not match the key type (e.g., a left encryption using a right public key) then the ciphertext is statistically independent of the message. Messy mode is useful to guarantee security against a *corrupt receiver:* the messy trapdoor makes it easy to extract the receiver bit and to create a fake ciphertext for the message which should not be transferred. On the other hand, in *decryption mode*, the system parameters are generated together with a *decryption trapdoor* which can be used to decrypt both left and right ciphertexts. Moreover, in decryption mode, left public keys are statistically indistinguishable from right public keys. Decryption mode is useful to guarantee security against a *corrupt sender:* the decryption trapdoor is used to create a public key which completely hides the receiver's selection bit, and to compute a decryption trapdoor and extracting both of the sender's messages. In each mode, the security of one party (i.e., the sender in messy mode, and the receiver in decryption mode) is guaranteed information theoretically. To achieve security for both parties simultaneously all that is needed is one simple computational requirement: the system parameters generated in messy mode need to be computationally indistinguishable from those generated in decryption mode.

3.2 Semi-adaptively Secure OT

In order to make the PVW OT protocol adaptively secure using our methodology, we need to make it semi-adaptively secure (Section 2.4). We do so by a series of simple transformations.

First, we observe that in the PVW protocol, the simulator must choose the CRS crs_{ot} based on which party is corrupt – i.e. the CRS should be in messy mode to handle a corrupt receiver or in decryption mode to handle a corrupt sender. This is a problem for us since the definition of semi-adaptive security requires that the simulator be setup-adaptive which means that it must simulate the CRS independently of any information on which parties are corrupted. We solve this issue by using a coin-tossing protocol to choose the CRS of the PVW OT protocol. Of course, coin-tossing requires the use of a UC secure commitment scheme which also needs its own CRS (crs_{com})! However, if we use an (efficient) adaptively secure commitment scheme (e.g., [12,10]) then the simulator's choice of crs_{com} can be independent of which party is corrupted. Unfortunately, this approach only works if the CRS for the OT protocol comes from a uniform distribution (over some group) and this also is not the case in all instantiations of the PVW protocol. However, we observe that the CRS of the OT protocol (crs_{ot}) can be divided into two parts $crs_{\mathrm{ot}} = (crs_{\mathrm{sys}}, crs_{\mathrm{tmp}})$, where a *system CRS* crs_{sys} can be independent of which party is corrupted (i.e., can be the same for both messy and decryption modes) but may not be uniform, while crs_{tmp} determines the mode and thus needs to depend on which party is corrupted, but this part is required to be uniform. Therefore we can use an ideal CRS functionality to choose the setup for our protocol which consists of ($crs_{\mathrm{com}}, crs_{\mathrm{sys}}$) and then run a coin-flipping protocol to choose the uniform crs_{tmp}.

Secondly, we must now consider the cases where one party is corrupted from the beginning, but the second party becomes corrupted adaptively during the protocol execution. Let us first consider the case where the sender starts out corrupted. In this case, to handle the corrupt sender, the simulator needs to simulate the execution in decryption mode. Moreover, to extract the sender's value, the simulator uses the decryption trapdoor to create a *dual public key* (on behalf of the receiver) which comes with both a left and a right secret key. Later, if the receiver becomes corrupted, the simulator needs to explain the randomness used by the receiver during key generation to create such a public key. Luckily, current dual-mode schemes already make this possible and we just update the definition with a property called *encryption key duality* to capture this.

Now, consider the case where the receiver is corrupted at the beginning but the sender might *also* become corrupted later on. In this case the simulator simulates the execution in messy mode. In particular, the simulator uses the messy trapdoor to identify the receiver key type (right or left) and thus extracts the receiver bit. Then the simulator learns the appropriate sender message for that bit and (honestly) produces the ciphertext for that message. In addition, the simulator must produce a "fake" ciphertext for the other message. Since, in messy mode, this other ciphertext is statistically independent of the message, it is easy to do so. However, if the sender gets corrupted later, the simulator must *explain* the fake ciphertext as an encryption of some particular message. To capture this ability, we require the existence of *internal state reconstruction* algorithm which can explain the fake ciphertext as an encryption of any message.

Again, we notice that the QR instantiation of the PVW scheme already satisfies this new notion as well.

We now specify our enhanced version of dual-mode encryption in more detail. Here we just describe the added features with respect to [28]; refer to [15] for the full description.

Enhanced Dual-Mode Encryption. A dual-mode cryptosystem for message space $\{0,1\}^n$ is defined by the following polynomial-time algorithms:

- $(crs, \tau) \leftarrow \mathsf{PG}(1^\lambda, \mu)$. The parameter generation algorithm PG is a randomized algorithm which takes security parameter λ and mode $\mu \in \{\text{mes}, \text{dec}\}$ as input, and outputs (crs, τ), where crs is a common reference string and τ is the corresponding trapdoor information. Note that PG includes two stages, PG_{sys} and PG_{tmp}, i.e., compute $(G, crs_{\text{sys}}, \tau_{\text{sys}}) \leftarrow \mathsf{PG}_{\text{sys}}(1^\lambda)$ and $(crs_{\text{tmp}}, \tau_{\text{tmp}}) \leftarrow \mathsf{PG}_{\text{tmp}}(\mu, G, crs_{\text{sys}}, \tau_{\text{sys}})$ where G is a group with operator "+", and set $crs \leftarrow (crs_{\text{sys}}, crs_{\text{tmp}})$ and $\tau \leftarrow (\tau_{\text{sys}}, \tau_{\text{tmp}})$. Also note that the system CRS is independent of mode μ.
- $(pk, sk) \leftarrow \mathsf{KG}(crs, \sigma)$; $(c, \zeta) \leftarrow \mathsf{Enc}(crs, pk, b, m)$; $m \leftarrow \mathsf{Dec}(crs, pk, sk, c)$; and $\rho \leftarrow \mathsf{MessyId}(crs, \tau, pk)$ as in [28].
- $(c, \omega) \leftarrow \mathsf{FakeEnc}(crs, \tau, pk, \rho)$. The fake encryption algorithm $\mathsf{FakeEnc}$ is a randomized algorithm. For the messy branch ρ, the ciphertext c is faked by using the trapdoor τ, and some internal information ω is saved for reconstructing the random coins used for encryption.
- $\zeta \leftarrow \mathsf{Recons}(crs, \tau, pk, \rho, c, \omega, m)$. The internal state reconstruction algorithm Recons is a deterministic algorithm. When the plaintext m is supplied for the faked ciphertext c in messy branch ρ, the algorithm recovers the used random coins ζ based on previously generated internal information ω.
- $(pk, sk_0, sk_1) \leftarrow \mathsf{DualKG}(crs, \tau)$. The dual key generation algorithm DualKG is a randomized algorithm, which based on the trapdoor τ, outputs an encryption key pk, and two decryption keys sk_0, sk_1 corresponding to key type 0 and 1, respectively.

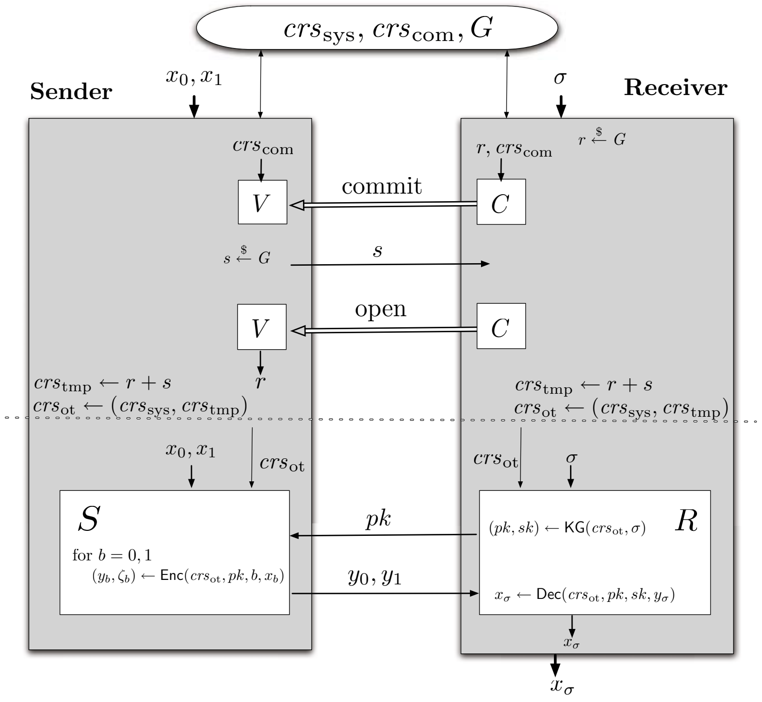
Definition 6 (*Enhanced* Dual-Mode Encryption). *An* enhanced dual-mode cryptosystem *is a tuple of algorithms as described above satisfying the following properties:*

- COMPLETENESS *as in* [28].
- ENHANCED MODE INDISTINGUISHABILITY: *The CRSes generated by* PG *in messy mode and in decryption mode are indistinguishable in the sense that (i) the both system CRSes are identically distributed, and (ii) the two temporal CRSes are computationally indistinguishable from random elements in group G.*
- MESSY BRANCH IDENTIFICATION AND CIPHERTEXT EQUIVOCATION: *For every* $(crs, \tau) \leftarrow \mathsf{PG}(1^\lambda, \text{mes})$ *and every pk,* $\mathsf{MessyId}(crs, \tau, pk)$ *outputs a branch value ρ such that for every $m \in \{0,1\}^n$,* $\mathsf{Enc}(crs, pk, \rho, \cdot)$ *is simulatable.*

– Encryption key duality: *For every* $(crs, \tau) \leftarrow \mathsf{PG}(1^\lambda, \text{dec})$*, there exists* $(pk, sk_0, sk_1) \leftarrow \mathsf{DualKG}(crs, \tau)$ *such that for every* $\sigma \in \{0,1\}$*,* (pk, sk_σ) *is statistically indistinguishable from the honestly generated key pair.*

Construction. Based on the above transformations, a generic construction for a semi-adaptively secure OT protocol is given in Figure 4. It consists of two phases, the coin tossing phase and the transferring phase (which is separated by a dot line in the figure). The CRS consists of two pieces: the first piece is a system CRS denoted as crs_{sys}, while the second piece is for an adaptively secure UC commitment protocol which will be used for constructing a coin tossing protocol. The UC commitment includes two stages, the commit (to a randomly selected value r by the receiver) and the open stages, which could be interactive, and is used to compute a temporal CRS crs_{tmp}. crs_{tmp} together with the system CRS crs_{sys} are used as the CRS for the transferring phase and we denote it as crs_{ot}. With crs_{ot} in hand, we "plug in" the PVW protocol, but based on the enhanced dual-mode cryptosystem to achieve message transferring.

Theorem 4. *Given an adaptively UC-secure commitment scheme and an enhanced dual-mode cryptosystem as in Definition 6, the protocol in Figure 4 semi-adaptively realizes* $\mathcal{F}_{\text{OT}}$ *in the* $\mathcal{F}_{\text{CRS}}$*-hybrid model.*



$crs_{\text{sys}}, crs_{\text{com}}, G$
Sender
Receiver
x_0, x_1
σ
crs_{com}
r, crs_{com}
$r \xleftarrow{\$} G$
V
C
commit
$s \xleftarrow{\$} G$
s
open
r
$crs_{\text{tmp}} \leftarrow r + s$
$crs_{\text{ot}} \leftarrow (crs_{\text{sys}}, crs_{\text{tmp}})$
crs_{ot}
S
for $b = 0, 1$
$(y_b, \zeta_b) \leftarrow \mathsf{Enc}(crs_{\text{ot}}, pk, b, x_b)$
pk
y_0, y_1
R
$(pk, sk) \leftarrow \mathsf{KG}(crs_{\text{ot}}, \sigma)$
$x_\sigma \leftarrow \mathsf{Dec}(crs_{\text{ot}}, pk, sk, y_\sigma)$
x_σ

Fig. 4. Generic semi-adaptively secure OT protocol

By "plugging in" efficient instantiations of the two building blocks above, we obtain efficient concrete protocols for semi-adaptively secure OT. For example, good candidates for adaptively secure UC commitments can be found in [10,12], while a QR-based dual-mode encryption scheme is presented in [28]; in the full version we show that this scheme also satisfies Definition 6. As mentioned in Section 1, a semi-adaptively secure OT protocol can also be based on the DDH assumption. In this case, however, in order to make ciphertext equivocation possible, we also need an efficient Σ-protocol for the equality of discrete logs.

3.3 Efficient and Adaptively Secure OT

We now apply our compiler from Section 2.4 to the protocol in Figure 4, to immediately obtain efficient adaptively secure OT protocols in the UC framework.

Corollary 5. *Assume that the DDH, QR, and DCR assumptions hold. Then there exists an adaptively secure protocol that UC-realizes the* bit-*OT functionality $\mathcal{F}_{\mathrm{OT}}$ in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid world, running in (expected) constant number of rounds and using (expected) constant number of public-key operations.*

Justification for the assumptions is as follows: efficient adaptive UC commitments can be realized in the CRS model under the DCR assumption [12], non-committing and somewhat non-committing encryption can be constructed under DDH ([11] and Section 2, respectively), while enhanced dual-model encryption exists under the QR assumption ([28] and Section 3.2).

In the full version we also show how to instantiate our framework using the DDH version of the PVW protocol, resulting in an efficient bit-OT protocol with similar parameters to the one above based on DCR. Further, we also show how to use the DDH version of PVW to also efficiently implement *string*-OT. This involves the semi-adaptively secure realization of an enhanced, *receiver-committed* version of bit-OT, where the receiver is also committed to his bit under an equivocal commitment scheme (e.g., a Pedersen commitment), as well as a generalization of our compiler; lastly, we use several copies of the enhanced bit-OT functionality to construct string-OT for long strings. (See [15] for details.) This strategy yields the following theorem.

Theorem 6. *Assume that the DDH and DCR assumptions hold. Then there exists an adaptively secure protocol that UC-realizes the* string-OT *functionality $\mathcal{F}_{\mathrm{OT}}$ in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid world, and can transfer an n-bit string in (strict) constant number of rounds and using (strict) $O(n)$ public-key operations.*

Acknowledgements. We thank Ran Canetti, Yevgeniy Dodis, Yuval Ishai, Stas Jarecki, and Aggelos Kiayias for useful discussions. We also thank the anonymous referees for their constructive comments.

References

1. Beaver, D.: Plug and play encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997)
2. Beaver, D.: Adaptively secure oblivious transfer. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 300–314. Springer, Heidelberg (1998)
3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (December 2005)
5. Canetti, R., Damgård, I., Dziembowski, S., Ishai, Y., Malkin, T.: Adaptive versus non-adaptive security of multi-party protocols. J. Cryptology 17(3), 153–207 (2004)
6. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
7. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503. ACM, New York (2002)
9. Choi, S.G., Dachaman-Soled, D., Malkin, T., Wee, H.: Simple, black-box constructions of adaptively secure protocols. In: Reingold, O. (ed.) TCC. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
10. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: STOC, pp. 426–437 (2003)
11. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
12. Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
13. Damgård, I., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
14. Damgård, I., Nielsen, J.B., Orlandi, C.: Essentially optimal universally composable oblivious transfer. In: ICISC, pp. 318–335 (2008)
15. Garay, J., Wichs, D., Zhou, H.-S.: Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. Cryptology ePrint Archive: Report 2008/534 (2008)
16. Garay, J.A., MacKenzie, P., Yang, K.: Efficient and universally composable committed oblivious transfer and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 297–316. Springer, Heidelberg (2004)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
18. Haitner, I.: Semi-honest to malicious oblivious transfer – the black-box way. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008)
19. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: STOC, pp. 99–108. ACM, New York (2006)

20. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
21. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
22. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
23. Lindell, A.Y.: Efficient fully-simulatable oblivious transfer. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008)
24. Lindell, Y.: Adaptively secure two-party computation with erasures. In: Fischlin, M. (ed.) CT-RSA. LNCS, vol. 5473, pp. 117–132. Springer, Heidelberg (2009)
25. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
26. Lindell, Y., Zarosim, H.: Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. In: Reingold, O. (ed.) TCC. LNCS, vol. 5444, pp. 183–201. Springer, Heidelberg (2009)
27. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
28. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)

Collusion-Free Multiparty Computation in the Mediated Model

Joël Alwen[1], Jonathan Katz[2], Yehuda Lindell[3], Giuseppe Persiano[4], abhi shelat[5], and Ivan Visconti[4,⋆]

[1] New York University, USA
jalwen@cs.nyu.edu
[2] The University of Maryland, USA
jkatz@cs.umd.edu
[3] Bar-Ilan University, Israel
lindell@cs.biu.ac.il
[4] University of Salerno, Italy
{giuper,visconti}@dia.unisa.it
[5] University of Virginia, USA
shelat@virginia.edu

Abstract. *Collusion-free* protocols prevent subliminal communication (i.e., covert channels) between parties running the protocol. In the standard communication model, if one-way functions exist, then protocols satisfying any reasonable degree of privacy cannot be collusion-free. To circumvent this impossibility, Alwen, shelat and Visconti (CRYPTO 2008) recently suggested the *mediated model* where all communication passes through a mediator. The goal is to design protocols where collusion-freeness is guaranteed as long as the mediator is honest, while standard security guarantees hold if the mediator is dishonest. In this model, they gave constructions of collusion-free protocols for commitments and zero-knowledge proofs in the two-party setting.

We strengthen the definition of Alwen et al., and resolve the main open questions in this area by showing a collusion-free protocol (in the mediated model) for computing any multi-party functionality.

1 Introduction

It is well known that two or more parties running some protocol can potentially embed "disallowed" communication in the protocol messages themselves; i.e., the parties can use the messages of the protocol as a *covert channel* to communicate in a subliminal (a.k.a., steganographic) fashion. As introduced by Lepinski, Micali, and shelat, a *collusion-free* protocol [13] rules out such covert

⋆ The research of the second and third authors was supported by US-Israel Binational Science Foundation grant #2004240. The research of the fifth and sixth authors has been supported in part by the European Commission through the ICT program under Contract ICT-2007-216646 ECRYPT II and through the FP6 program under contract FP6-1596 AEOLUS. Author shelat is supported by NSF grant 0845811.

communication. Unfortunately, in the standard communication model, if one way functions exist, then it is impossible for any protocol whose messages have any entropy to be collusion-free [9]. This seems to rule out collusion-free protocols in the standard communication model that realize any "interesting" level of privacy [13].

Although there has been some work addressing the issue of subliminal channels in certain limited contexts (mainly signature schemes [3,4,7,17,19]), the problem has, until recently, been largely ignored by the cryptographic community. Presumably this is because protocol designers generally assume a "worst-case" adversarial model, where if two parties are dishonest then they are assumed to be coordinating their actions and communicating out of band anyway. Recent attention focused on applying cryptographic protocols in game-theoretic settings [10,11,13] (see also [12]), however, has re-invigorated interest in designing collusion-free protocols. Preventing subliminal communication is also important in other settings. For example, in a large-scale, distributed system where parties are chosen randomly (from a large pool of players) to run some protocol, the set of parties running a given instance of the protocol may not have had any chance to coordinate their actions in advance, and may have no way to communicate out of band; in this case, the protocol itself introduces a new vulnerability if it can be used as a means for players to initiate collusion, or to transfer information. The problem of subliminal communication is not just of theoretical interest: efforts to collude using covert channels have been observed in high-profile spectrum auctions [6].

One approach for constructing collusion-free protocols is to rely on the notion of *verifiable determinism* as introduced by Lepinski et al. [12,13]. Roughly speaking, verifiable determinism ensures that at every point in the protocol there is only a *single* "valid" message that a player can send; if that player sends anything else, all other parties detect this and raise an alarm. This suffices to prevent covert communication. Unfortunately, all existing constructions of verifiably deterministic protocols for general secure computation [10,11,13] rely on strong physical assumptions such as secure envelopes and ballot boxes.

A completely different approach to the problem was recently suggested by Alwen, shelat and Visconti [1]. They proposed a model in which each party is able to communicate only with a *mediator*. (I.e., the communication network is a star graph with the mediator at the center.) Rather than *remove* randomness from protocol messages, as when using verifiable determinism, this approach has the mediator *add* randomness to (i.e., re-randomize) the messages of the protocol in order to eliminate any subliminal communication. This, of course, assumes the mediator is honest; when the mediator is dishonest then corrupted parties can communicate freely using the mediator as a channel. In this case, the protocol is required to satisfy standard security guarantees.

The mediated model can be realized in many real settings. As an example, recently in Israel the Maccabi Health Fund (a large HMO) ran an auction with several insurance companies as bidders. In this auction, the bidders came to the offices of the HMO and were seated in separate rooms, with no way to communicate with the outside world (participants were searched for cellphones and other wireless devices). The auction proceeded in stages with an auctioneer going from room to room, informing the participants about the results of the previous round and taking their next bid. It would have been possible in this case to replace the auctioneer with a server mediating the communication between all parties.

1.1 Our Contributions

In addition to introducing a definition of collusion-freeness in the mediated model, Alwen et al. also gave the first constructions of collusion-free protocols in this setting. They showed protocols for commitment and zero-knowledge in the two-party case, but left open the questions of general secure computation as well as dealing with more than two parties. In this paper we solve these open questions, and show the first multi-party protocol for collusion-free computation of arbitrary functionalities in the mediated model. Feasibility is not trivial in this setting, in part because we aim to satisfy a *stronger* definition of security than that put forth by Alwen et al.; see below. (We view this strengthened definition as an additional contribution of our work.) Finally, we prove composition theorems in the mediated setting that may be useful in future work.

The paragraphs that follow briefly describe the most important differences between our definition and that of Alwen et al. [1]; formal definitions are in Section 2. The next few paragraphs provide a high-level overview of our protocol that emphasizes the technical difficulties that arise.

Aborts as a subliminal channel. The definition in [1] allows parties to communicate some (bounded) number of bits by *aborting* the protocol; specifically, in an r-round protocol each party can communicate roughly $\log r$ bits to all other parties. Alwen et al. conjecture that this is unavoidable. We show that this conjecture is *false*. In our definition we allow only a *single* bit to be communicated, where this bit indicates whether some party aborted at some point in the protocol but does not reveal which parties aborted or in which rounds these aborts occurred. Achieving this stronger notion introduces many of the complications in designing our protocol.

Set-up assumptions. Alwen et al. assume *no* shared state between the parties, and this allows the mediator to potentially "fork" the parties (in the sense of [2]) into disjoint subsets running independent computations. To prevent this, Alwen et al. assume that even a dishonest mediator behaves honestly during a "set-up" phase. (See further discussion in Section 2.)

In addition to this model, we also analyze a model where there is assumed to be a trusted public-key infrastructure (PKI) such that all parties running the protocol know each others' public keys. These two set-up assumptions are incomparable: the first makes assumptions regarding the behavior of the mediator but can achieve a stronger notion of collusion-freeness; the second may be more realistic but requires the involvement of an external trusted party to set up the public key infrastructure.

1.2 Overview of Our Protocol

The discussion here omits certain details and is meant only to illustrate the high-level structure of the protocol. A formal description of the protocol is given in Section 3.

Let $P_1, \ldots, P_n$ be a set of n parties, each communicating with a mediator P_{n+1}, who wish to compute some (randomized) functionality $\mathcal{F}$. Let π be a protocol that securely computes $\mathcal{F}$ in the standard communication model with broadcast. (In fact, we assume without loss of generality that all messages in π are sent over the broadcast channel.) We compile π to obtain a collusion-free protocol Π in the following way. For each message msg sent by some party P_i in protocol π do:

1. P_i and the mediator run a protocol for secure two-party computation of a functionality $\mathcal{F}^{\pi}_{\mathsf{compute}}$ that outputs to the mediator the next message msg that P_i would send in the underlying execution of π. (A secure computation is needed since P_i will not actually know any of the messages sent by other parties in previous rounds of π; see step 2.)

   If the mediator does not obtain a valid msg (i.e., if P_i aborts or provides incorrect input to $\mathcal{F}^{\pi}_{\mathsf{compute}}$), then the mediator sets msg to some default value. (This step is essential if we wish to prevent parties from using aborts as a covert channel.)
2. The mediator sends independent *commitments* of msg to each of the other parties.

At the end of the protocol, the mediator runs a secure two-party computation with each party P_i that allows P_i to learn their output, as specified by protocol π.

It is not too difficult to argue that the above protocol is collusion-free when the mediator is honest. Intuitively, this is because each party sees only *independent commitments* to messages rather than the messages themselves. However, the following issues arise due to the need to preserve security when the mediator is dishonest:

Authentication. The mediator should be prevented from modifying the messages of honest parties. To achieve this, we change $\mathcal{F}^{\pi}_{\mathsf{compute}}$ to output (msg, σ), where σ is a valid signature[1] by P_i on msg, and require the mediator to send commitments on *both* these values to the other parties. Furthermore, $\mathcal{F}^{\pi}_{\mathsf{compute}}$ will ensure that all previous commitments contain appropriately signed messages.

Preventing subliminal channels based on aborts. Signing each message (as just described) prevents a dishonest mediator from modifying honest parties' messages, but introduces a potential problem with collusion-freeness when the mediator is honest: if a party aborts, the mediator has no way of generating a (commitment to a) default message with an appropriate signature. We fix this by allowing the mediator in this case to commit to a "dummy message" with

[1] As discussed earlier, we consider two different set-up assumptions: public keys can be established either during a preamble phase, or via an external PKI. See further discussion in the following section.

no signature; we also change $\mathcal{F}^{\pi}_{\mathsf{compute}}$ so that if it detects a dummy message the mediator receives no output. The effect is that parties cannot detect whether anyone has aborted until the end of the protocol, and never learn which (or how many) parties aborted nor the round(s) in which an abort occurred.

Ensuring "broadcast". Protocol π is secure under the assumption that parties communicate over a broadcast channel. In our compiled protocol, where all communication is routed through the mediator, we need a way to ensure that a dishonest mediator sends (different commitments to) the *same* message to all parties. We implement this "mediator broadcast" by, roughly speaking, having the mediator (1) collect signatures from all parties on the committed messages; (2) send independent commitments on these signatures to all parties; and then (3) prove to each party independently that all parties have signed a commitment to the same underlying message. As above, in case of an abort we allow the mediator to send a "dummy commitment" to the parties.

Handling concurrency. When the mediator is honest, the protocols computing $\mathcal{F}^{\pi}_{\mathsf{compute}}$, as well as the sub-protocols used to implement mediator broadcast, are run sequentially. But when the mediator is dishonest, it may run *concurrent* executions with the honest parties. We thus need all the two-party protocols being run to be secure under (bounded) concurrent self composition.

2 Definitions

Standard cryptographic primitives. Let C be a perfectly binding commitment scheme, where $C(m;r)$ denotes a commitment to m using random coins r. The decommitment of $\mathsf{com} = C(m;r)$ is $\mathsf{dec} = (m,r)$. We assume the length of all commitments is a fixed function of the message length.

Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme that is existentially unforgeable under adaptive chosen-message attacks. $\mathsf{Range}(\mathsf{Gen})$ denotes the set of outputs of Gen, and we assume (without loss of generality) that one can efficiently decide whether a given (sk, pk) lies in $\mathsf{Range}(\mathsf{Gen})$. We assume the length of all valid signatures is some known, fixed function of the message length.

Security in the mediated model – preliminaries. We use the real/ideal paradigm for defining security, but our real and ideal worlds differ from the usual ones and collusion-freeness requires a new definition. Our real world is essentially standard except that all communication is between parties $P_1, \ldots, P_n$ and the mediator P_{n+1}. We define two different ideal worlds depending on whether the mediator is honest (and collusion-freeness is the goal) or dishonest (in which case we default to the standard notion of security). In each ideal world we consider two possible set-up assumptions; see below. Collusion-freeness is defined by requiring the existence of *independent* simulators, one for each malicious party, such that their joint output in the ideal world is indistinguishable from the joint output of the malicious parties in the real world.

Let $\mathcal{F} = (f_1, \ldots, f_{n+1})$ denote the functionality the parties wish to compute, where each f_i maps $n+1$ inputs to a single output. (We allow the mediator to provide input and receive output, something not done in [1].) We implicitly assume that any protocol under discussion for computing $\mathcal{F}$ is *correct*: i.e., if parties $P_1, \ldots, P_{n+1}$ have inputs $x_1, \ldots, x_{n+1}$ and run the protocol honestly, then each P_i receives output $f_i(x_1, \ldots, x_{n+1})$, distributed appropriately in case $\mathcal{F}$ is randomized.

Set-up assumptions. Alwen et al. [1] observe that if *no* setup is assumed, then—because all parties communicate only with the mediator—a dishonest mediator can "fork" the parties into disjoint groups, each running a separate computation (much as in [2]). To prevent this, parties must be able to "authenticate" to each other. (It is as an interesting open question to assume no setup and treat "forking" attacks directly, but this is not the focus of our work.)

Here, we study two set-up assumptions under which such authentication can be implemented. The first, termed trusted-PKI, assumes a PKI in the usual sense, with each party knowing the public keys of all the other parties running the protocol. We stress that we do not assume honestly generated keys, or require parties to prove knowledge of their keys; all we require is consistency.

While this approach is appealing, it loses something in the "spirit" of collusion-freeness since parties are now potentially able to follow strategies based on each others' identities. Nevertheless, we believe the trusted-PKI model is meaningful in the context of collusion-freeness. For one, public keys can be generated *before* inputs are given to the parties (and before the function being computed is agreed upon!), and so the use of any subliminal communication will be limited. Furthermore, parties who are not aware of each other before execution of the protocol will necessarily generate their public keys independently, whereas parties who *are* aware of each other before executing the protocol cannot be prevented anyway from communicating arbitrary information in advance.

We also consider the mediated-PKI model that provides stronger guarantees of collusion-freeness under a different assumption. Specifically, here we follow Alwen et al. [1] and assume that even a dishonest mediator follows the protocol during a "preamble" phase where a "pseudo-PKI" is established. Instead of viewing this as an assumption, one can also interpret this as a claim that *if* the mediator acts in a particular way *then* certain guarantees hold.

2.1 Execution in the Real World (with an Honest Mediator)

We first consider the real world (i.e., the mediated model) in which an $(n+1)$-party protocol Π is executed. Channels are available only between the mediator P_{n+1} and P_i (for all i). For simplicity the channels to/from P_{n+1} are assumed to be private and authenticated.

In this section we assume the mediator is honest; we consider the case of a dishonest mediator in Section 2.4. Let $\mathcal{I} \subseteq [n]$ denote the set of corrupt parties[2] and denote by $\mathcal{H} = [n] \setminus \mathcal{I}$ the set of uncorrupted parties (not including the mediator). A real world execution begins with a "PKI establishment" stage for which we define two variants:

[2] In contrast to the usual case, here a meaningful definition is obtained even when $\mathcal{I} = [n]$.

– $\mathsf{trusted\text{-}PKI}$: For $i \in \mathcal{H}$, party P_i honestly generates a signature key pair (pk_i, sk_i). Each corrupted party P_i may generate an arbitrary public key pk_i. Once all keys have been generated, each P_i is given the vector $\mathsf{KeyVec}_i := (pk_1, \ldots, pk_n)$ of public keys, and the mediator is given $\mathsf{KeyVec}^i_{n+1} := \mathsf{KeyVec}_i$ for all $i \in [n]$. (The notation is chosen to be consistent with the $\mathsf{mediated\text{-}PKI}$ setting below.) We say that KeyVec_i *matches* KeyVec^i_{n+1} if they are equal.
– $\mathsf{mediated\text{-}PKI}$: For $i \in \mathcal{H}$, party P_i honestly generates a signature key pair (pk_i, sk_i). Each corrupted party P_i may generate an arbitrary public key pk_i. All parties send pk_i to the mediator. (If a party fails to send anything, the mediator uses a default public key.) The mediator chooses independent coins $\{r^j_i\}_{i,j\in[n]}$, computes $c^j_i = C(pk_i; r^j_i)$, and sends $\mathsf{KeyVec}_i := (c^i_1, \ldots, c^i_n)$ to party P_i. The mediator keeps the vectors of decommitments $\mathsf{KeyVec}^i_{n+1} := ((pk_1, r^i_1), \ldots, (pk_n, r^i_n))$. We say that KeyVec_i *matches* KeyVec^i_{n+1} if for all $j \in [n]$ the jth component of KeyVec^i_{n+1} is a valid decommitment to the jth component of KeyVec_i.

The remainder of the real-world execution is identical in both settings.

Input determination and protocol execution: Each party P_i (for $i \in [n+1]$) is given input x_i. Party P_i is also given auxiliary input aux_i (which honest players ignore) as well as independent random coins r_i. The parties then run the protocol, with honest parties (including the mediator) acting as directed by Π, and corrupted parties behaving arbitrarily.

Result of the experiment: At the conclusion of the protocol, let OUT_i, for $i \in \mathcal{I}$, denote the entire view of the corrupted party P_i, and let OUT_i, for $i \in \mathcal{H} \cup \{n+1\}$, denote the final output of P_i (as dictated by Π). Given a set of adversarial strategies $\mathcal{A}_\mathcal{I} = \{\mathcal{A}_i\}_{i\in\mathcal{I}}$, define

$$\text{REAL}^{\mathsf{mediated}}_{\Pi,\mathcal{A}_\mathcal{I}(\mathbf{aux})}(1^k, \boldsymbol{x}) \stackrel{\text{def}}{=} (\text{OUT}_1, \ldots, \text{OUT}_{n+1})$$

to be the random variable consisting of the stated outputs following an execution of Π where the parties are given inputs $\boldsymbol{x} = \{x_1, \ldots, x_{n+1}\}$ and auxiliary inputs $\mathbf{aux} = \{\mathsf{aux}_1, \ldots, \mathsf{aux}_{n+1}\}$.

2.2 Execution in the Ideal World (with an Honest Mediator)

We continue to assume the mediator is honest. In this ideal world, all parties communicate only with a trusted party computing $\mathcal{F}$. In particular, corrupted parties are unable to communicate with each other and therefore cannot communicate information about their inputs or coordinate their actions (beyond what they have agreed upon in advance). Let $\mathcal{I} \subseteq [n]$ be the set of corrupted parties, and let $\mathcal{H} = [n] \setminus \mathcal{I}$ be the set of honest parties (other than the mediator) as before.

As in the previous section, we distinguish between two settings in the ideal world. The $\mathsf{trusted\text{-}PKI}$ setting includes a "PKI establishment" step where a PKI is established exactly as in the real world: for $i \in \mathcal{H}$, party P_i honestly generates signature keys (pk_i, sk_i). A corrupted P_i outputs any public key pk_i of its choice. For $i \in [n]$ party P_i is then given the vector $\mathsf{KeyVec}_i := (pk_1, \ldots, pk_n)$ of public keys. The mediated-PKI setting has no PKI establishment step.

The remainder of the ideal-world execution is identical in both settings.

Input determination: Each party P_i (for $i \in [n+1]$) is given their input x_i and auxiliary input aux_i (which an honest player ignores).
An honest party sets $x'_i = x_i$ and sends x'_i to $\mathcal{F}$. A corrupted P_i may send any x'_i of its choice. Unless otherwise specified, if any $x'_i = \perp$ then all parties get output $\perp$ from $\mathcal{F}$. Otherwise, $\mathcal{F}$ hands $f_i(x'_1, \ldots, x'_{n+1})$ to party P_i, for $i \in [n+1]$.

Note that a malicious party who "aborts" by sending $\perp$ to $\mathcal{F}$ communicates (at most) one additional bit to all other parties beyond what is directly implied by $\mathcal{F}$. Furthermore, this decision to abort must be made independently of the output of $\mathcal{F}$ on the given inputs.

Result of the experiment: At the conclusion of the protocol, let OUT_i, for $i \in \mathcal{I}$, denote an arbitrary value output by P_i, and let OUT_i, for $i \in \mathcal{H} \cup \{n+1\}$, denote the value given to P_i by $\mathcal{F}$. Given a set of adversarial strategies $\mathcal{S}_\mathcal{I} = \{\mathcal{S}_i\}_{i \in \mathcal{I}}$, define

$$\text{IDEAL}^{\mathsf{cf}}_{\mathcal{F}, \mathcal{S}_\mathcal{I}(\mathbf{aux})}(1^k, \boldsymbol{x}) \stackrel{\text{def}}{=} (\text{OUT}_1, \ldots, \text{OUT}_{n+1})$$

to be the random variable consisting of the stated outputs following an ideal-world execution where the parties are given inputs $\boldsymbol{x}$ and auxiliary inputs $\mathbf{aux}$ as specified.

2.3 Collusion-Freeness

Having defined the ideal and real models, we can now define collusion-freeness. If we followed the standard definitional paradigm, we would require that for all $\mathcal{I}$ and any set of efficient real-world strategies $\mathcal{A}_\mathcal{I} = \{\mathcal{A}_i\}_{i \in \mathcal{I}}$, there should exist a set of efficient ideal-world strategies $\mathcal{S}_\mathcal{I} = \{\mathcal{S}_i\}_{i \in \mathcal{I}}$ such that the corresponding real- and ideal-world outcomes are computationally indistinguishable. A deficiency of this approach is that it allows each $\mathcal{S}_i$ to depend on $\mathcal{I}$ as well as *all* the $\mathcal{A}_j$ (i.e., even for $j \neq i$), and thus this approach does not adequately model collusion-freeness. Since we want each $\mathcal{S}_i$ to depend only on $\mathcal{A}_i$, we instead require the existence of a set of efficient *transformations* $\{\mathsf{Sim}_i\}_{i \in [n]}$ such that setting $\mathcal{S}_i = \mathsf{Sim}_i(1^k, \mathcal{A}_i)$ for $i \in \mathcal{I}$ makes the real and ideal worlds indistinguishable.

Definition 1. *Let $\mathcal{F}$ be a functionality, and Π an $(n+1)$-party protocol computing $\mathcal{F}$ in the mediated model. Π is a* collusion-free protocol computing *$\mathcal{F}$ if there is a set $\{\mathsf{Sim}_i\}_{i \in [n]}$ of efficiently-computable transformations such that, for all $\mathcal{I} \subseteq [n]$ and any* PPT *strategies $\{\mathcal{A}_i\}_{i \in \mathcal{I}}$, setting $\mathcal{S}_i = \mathsf{Sim}_i(1^k, \mathcal{A}_i)$ for $i \in \mathcal{I}$ implies that the following two distributions are computationally indistinguishable:*

$$\left\{\text{IDEAL}^{\mathsf{cf}}_{\mathcal{F}, \mathcal{S}_\mathcal{I}(\mathbf{aux})}(1^k, \boldsymbol{x})\right\}_{\boldsymbol{x}, \mathbf{aux} \in (\{0,1\}^*)^{n+1},\, k \in \mathbb{N}}$$

$$\left\{\text{REAL}^{\mathsf{mediated}}_{\Pi, \mathcal{A}_\mathcal{I}(\mathbf{aux})}(1^k, \boldsymbol{x})\right\}_{\boldsymbol{x}, \mathbf{aux} \in (\{0,1\}^*)^{n+1},\, k \in \mathbb{N}}$$

2.4 Security (with a Dishonest Mediator)

The definition in the case of a dishonest mediator is essentially the standard one for secure multi-party computation, with the exception being that honest parties cannot communicate directly in the real world. (We also incorporate a PKI establishment phase as in the prior sections.) Further details are given in the full version. A protocol satisfying both Definition 1 and the definition for the case of a dishonest mediator will be called a collusion-free protocol securely computing $\mathcal{F}$.

3 Collusion-Free Multiparty Computation in the Mediated Model

We construct a collusion-free protocol Π for secure computation of an arbitrary (poly-time) functionality $\mathcal{F} = (f_1, \ldots, f_{n+1})$. We first introduce the components of our protocol, and describe the protocol in full detail in Section 3.4. High-level intuition for the protocol was given in Section 1.2.

3.1 Building Blocks

Our protocol uses some cryptographic primitives and tools which we review here.

Two-party functionalities. We use ideal functionalities to model various sub-protocols used in Π. Standard functionalities we use are the commitment functionality $\mathcal{F}_{\mathsf{com}}$, the coin-tossing functionality $\mathcal{F}_{\mathsf{ct}}$, the zero-knowledge functionality $\mathcal{F}_{\mathsf{zk}}$, and the signature functionality $\mathcal{F}_{\mathsf{Sign}}$:

1. $\mathcal{F}_{\mathsf{com}}$ is defined by $\mathcal{F}_{\mathsf{com}}((m, r), \lambda) = (\perp, C(m; r))$, where λ denotes the empty string.
2. The coin-tossing functionality is defined by $\mathcal{F}_{\mathsf{ct}}(1^\ell, \lambda) = ((r, s), C(r; s))$, where $|r| = \ell$ and both r and s are uniformly distributed.
3. Let R be an $\mathcal{NP}$-relation. Functionality $\mathcal{F}_{\mathsf{zk}}$ for the relation R is defined by

$$\mathcal{F}_{\mathsf{zk}}((x, w), x') = \begin{cases} (\perp, R(x, w)) & \text{if } x = x' \\ (\perp, 0) & \text{otherwise} \end{cases}$$

4. The signature functionality is defined as:

$$\mathcal{F}_{\mathsf{Sign}}((sk, pk, m), (pk', m')) = \begin{cases} (\perp, \mathsf{Sign}_{sk}(m)) & \text{if } (pk, m) = (pk', m') \text{ and} \\ & \qquad (sk, pk) \in \mathsf{Range}(\mathsf{Gen}) \\ (\perp, \perp) & \text{otherwise} \end{cases}$$

A protocol π securely computing $\mathcal{F}$ (in the standard sense): Let π be an $(n+1)$-party protocol that securely computes $\mathcal{F}$ in the usual sense [8], in the standard communication model where all parties have access to a public (but authenticated) broadcast channel. Precisely, π is secure-with-designated-abort for any number $t \leq n+1$ of corrupted parties, where the mediator P_{n+1} is designated as the party who can prematurely abort the protocol. Roughly speaking, this means that the protocol guarantees privacy and correctness regardless of how many parties are corrupted, and guarantees output delivery and complete fairness as long as the mediator is not corrupted. For technical reasons, we also assume that π is proved secure using a *black-box* simulator.

By using standard techniques, we may assume without loss of generality that:

- All messages in π have the same, fixed length. In any given round only a single party broadcasts, and the identity of the party who broadcasts depends on the current round number only.
- Say π has r rounds. Then P_{n+1} learns its output in round $r-1$; party P_{n+1} broadcasts in round r; and every other party learns its output in round r.

Dummy commitments: As described in Section 1.2, everything the mediator sends to the parties will be "wrapped" inside a commitment. When all parties behave honestly, these will all be commitments to legitimate messages of protocol π along with a digital signature. If some party P_i aborts (or otherwise deviates from the protocol), however, an honest mediator will not be able to generate a valid commitment of this sort (in particular, the mediator will be unable to forge an appropriate signature). Nevertheless, we do not want some other party to learn that P_i aborted the protocol Π. We achieve this by allowing the mediator to send special "dummy commitments" to a distinguished value dummy. (I.e., a dummy commitment takes the form $C(\mathsf{dummy}; r)$.) For the sake of concreteness, dummy can be taken to be a string of 0s of the appropriate length if we require that all legitimate messages be prefixed by a '1'.

3.2 Oblivious Computation of π

The general structure of protocol Π, as described in Section 1.2, has the mediator send to each P_j *commitments* to all the protocol messages of π. Thus, P_j cannot compute its π-messages directly (since it cannot directly observe the π-messages of other parties), but must instead compute these messages by executing a two-party protocol with the mediator. Specifically, we define a functionality $\mathcal{F}^{\pi}_{\mathsf{compute}}$ that computes the next π-message of P_j along with a signature of P_j on that message, and a functionality $\mathcal{F}^{\pi}_{\mathsf{output}}$ that enables P_j to obtain its π-output. (The actual functionalities we require are more complex because we must also check for incorrect behavior on the part of the mediator.) These are defined formally in Figures 1 and 2. Observe that only the mediator P_{n+1} receives output from $\mathcal{F}^{\pi}_{\mathsf{compute}}$, and only P_j receives output from $\mathcal{F}^{\pi}_{\mathsf{output}}$.

3.3 Mediator Broadcast

Protocol π assumes that all parties communicate over a broadcast channel. When the mediator is corrupt, we therefore must ensure that the mediator sends (commitments to) the *same* message to all honest parties. Note that checking for signatures on protocol messages, as done by $\mathcal{F}^{\pi}_{\mathsf{compute}}$ and $\mathcal{F}^{\pi}_{\mathsf{output}}$, only ensures that this holds for the messages of *honest* parties; it does not prevent a dishonest mediator

Functionality $\mathcal{F}^{\pi}_{\mathsf{compute}}$

Functionality $\mathcal{F}^{\pi}_{\mathsf{compute}}$ runs with two parties P_j and P_{n+1} and works as follows:

- P_j inputs a pair of commitments $\mathsf{com}^{\mathsf{input}}$ and $\mathsf{com}^{\mathsf{rand}}$; a vector of commitments $\boldsymbol{C}$; vector KeyVec; and a round number rid. In addition, P_j sends two strings $\mathsf{dec}^{\mathsf{input}}$ and $\mathsf{dec}^{\mathsf{rand}}$, and its signing key sk_j.
- P_{n+1} inputs a pair of commitments $\mathsf{com}^{\mathsf{input}}_j$ and $\mathsf{com}^{\mathsf{rand}}_j$; a vector of commitments $\boldsymbol{C}_j$; vector KeyVec'; and a round number rid'. In addition, P_{n+1} sends a vector $\mathbf{dec}_j$.
- Upon receiving the above, $\mathcal{F}^{\pi}_{\mathsf{compute}}$ does:
  1. If KeyVec does not match KeyVec' then send $\perp$ to P_{n+1} and halt. Otherwise extract the parties' public keys $(pk_1, \dots, pk_n)$ from KeyVec'.
  2. If $(\mathsf{com}^{\mathsf{input}}, \mathsf{com}^{\mathsf{rand}}, \boldsymbol{C}, \mathsf{rid}) \neq (\mathsf{com}^{\mathsf{input}}_j, \mathsf{com}^{\mathsf{rand}}_j, \boldsymbol{C}_j, \mathsf{rid}')$ or if $(sk_j, pk_j) \notin \mathsf{Range}(\mathsf{Gen})$, then send $\perp$ to P_{n+1} and halt.
  3. If $\mathsf{dec}^{\mathsf{input}}$ is not a valid decommitment to $\mathsf{com}^{\mathsf{input}}_j$, or $\mathsf{dec}^{\mathsf{rand}}$ is not a valid decommitment to $\mathsf{com}^{\mathsf{rand}}_j$, or $\mathbf{dec}_j$ does not contain valid decommitments to all the commitments in $\boldsymbol{C}_j$, then send $\perp$ to P_{n+1} and halt.
  4. Let $(\mathsf{msg}_1, \sigma_1), \dots, (\mathsf{msg}_\ell, \sigma_\ell)$ be the committed values in $\boldsymbol{C}_j$. If any of these are dummy values, send $\perp$ to P_{n+1} and halt. For $1 \leq i \leq \ell$, let ℓ_i denote the index of the party who is supposed to broadcast in round i of π. If there exists an i such that (1) $\ell_i \neq n+1$ and (2) $\mathsf{Vrfy}_{pk_{\ell_i}}((\mathsf{msg}_i, 0i), \sigma_i) \neq 1$, then send $\perp$ to P_{n+1} and halt.
  5. Let x_j and r_j be the committed values in $\mathsf{com}^{\mathsf{input}}_j$ and $\mathsf{com}^{\mathsf{rand}}_j$ respectively. Compute the next message msg that party P_j would send in protocol π when running with input x_j, random tape r_j, and after receiving messages $\mathsf{msg}_1, \dots, \mathsf{msg}_\ell$. In addition, compute $\sigma = \mathsf{Sign}_{sk_j}(\mathsf{msg}, \mathsf{rid})$. Send (msg, σ) to P_{n+1} and halt.

Fig. 1. The functionality computing the next message of π

from sending different messages on behalf of *corrupted* parties (who may collude with the mediator and sign multiple messages).

We achieve the above using what we call "mediator broadcast." The mediator P_{n+1} begins holding a message m, and at the end of the protocol each P_i obtains an (independent) commitment com_i to a message m_i. The desired functionality is, informally, as follows: If all parties are honest, then $m_i = m$ for all P_i. If P_{n+1} is honest, then there is an $m' \in \{m, \mathsf{dummy}\}$ such that $m_i = m'$ for all honest parties P_i. If P_{n+1} is dishonest, then there is an m' such that $m_i \in \{m', \mathsf{dummy}\}$ for all honest parties P_i. This is a weak form of broadcast, but suffices for our application.

In Figure 3, we formally define a functionality $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$, parameterized by a session id sid, implementing the above. (An honest mediator chooses $r_1, \dots, r_n$ uniformly at random, and sets $\mathcal{H} = [n]$; an honest P_i sends $b_i = 1$.) We stress that the functionality *always* outputs a commitment for each party, *even if some (dishonest) party aborts*. Our protocol $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$ realizing $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$ proceeds, roughly speaking in the following three stages:

Functionality $\mathcal{F}^{\pi}_{\mathsf{output}}$

Functionality $\mathcal{F}^{\pi}_{\mathsf{output}}$ runs with two parties P_j and P_{n+1} and works as follows:

– P_j inputs a pair of commitments $\mathsf{com}^{\mathsf{input}}$ and $\mathsf{com}^{\mathsf{rand}}$, vector KeyVec, and a vector of r commitments $\boldsymbol{C}$. In addition, P_j sends two strings $\mathsf{dec}^{\mathsf{input}}$ and $\mathsf{dec}^{\mathsf{rand}}$.
– P_{n+1} inputs a pair of commitments $\mathsf{com}_j^{\mathsf{input}}$ and $\mathsf{com}_j^{\mathsf{rand}}$, vector KeyVec', and a vector of r commitments $\boldsymbol{C}_j$. In addition, P_{n+1} sends a vector $\mathbf{dec}_j$.
– Upon receiving the above, $\mathcal{F}^{\pi}_{\mathsf{output}}$ does:
  1. If KeyVec does not match KeyVec' then send $\perp$ to P_j and halt. Otherwise extract the parties' public keys $(pk_1, \ldots, pk_n)$ from KeyVec'.
  2. If $(\mathsf{com}^{\mathsf{input}}, \mathsf{com}^{\mathsf{rand}}, \boldsymbol{C}) \neq (\mathsf{com}_j^{\mathsf{input}}, \mathsf{com}_j^{\mathsf{rand}}, \boldsymbol{C}_j)$, then send $\perp$ to P_j and halt.
  3. If $\mathsf{dec}^{\mathsf{input}}$ (resp., $\mathsf{dec}^{\mathsf{rand}}$) is not a valid decommitment to $\mathsf{com}_j^{\mathsf{input}}$ (resp., $\mathsf{com}_j^{\mathsf{rand}}$), or $\mathbf{dec}_j$ does not contain valid decommitments to all the commitments in $\boldsymbol{C}_j$, then send $\perp$ to P_j and halt.
  4. Let $(\mathsf{msg}_1, \sigma_1), \ldots, (\mathsf{msg}_r, \sigma_r)$ be the committed values in $\boldsymbol{C}_j$. If any of these are dummy values, send $\perp$ to P_j and halt. For $1 \leq i \leq r$, let ℓ_i denote the index of the party who is supposed to broadcast in round i of π. If there exists an i such that (1) $\ell_i \neq n+1$ and (2) $\mathsf{Vrfy}_{pk_{\ell_i}}((\mathsf{msg}_i, 0i), \sigma_i) \neq 1$, then send $\perp$ to P_j and halt.
  5. Let x_j and r_j be the committed values in $\mathsf{com}_j^{\mathsf{input}}$ and $\mathsf{com}_j^{\mathsf{rand}}$. Compute the value $\textsc{out}_j$ that party P_j would output in protocol π when running with input x_j, random tape r_j, and after receiving the messages $\mathsf{msg}_1, \ldots, \mathsf{msg}_r$. Send $\textsc{out}_j$ to P_j and halt.

Fig. 2. The functionality computing the output of π

$\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$

Functionality $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$ runs with $P_1, \ldots, P_n, P_{n+1}$ as follows:

- For $j \in [n]$, each P_j inputs a bit b_j and KeyVec_j.
- P_{n+1} inputs a message m, $\{\mathsf{KeyVec}^j_{n+1}\}_{j\in[n]}$, random coins $r_1, \ldots, r_n$, and a set $\mathcal{H} \subseteq [n]$.
- For all $i \in [n]$, if KeyVec_i and KeyVec^i_{n+1} do not match then set $\mathcal{H} := \mathcal{H} \setminus \{i\}$.
- Let $b = \bigwedge_i b_i$.
- If $b = 1$, then:
  - For $i \in \mathcal{H}$ send $\mathsf{com}_i = C(m; r_i)$ to P_i.
  - For $i \in [n] \setminus \mathcal{H}$, send $\mathsf{com}_i = C(\mathsf{dummy}; r_i)$ to P_i.

  If $b = 0$, then for $i \in [n]$ send $\mathsf{com}_i = C(\mathsf{dummy}; r_i)$ to P_i. In either case, send b to P_{n+1}.

Fig. 3. Mediator broadcast

1. P_{n+1} sends $\mathsf{com}_i = C(m; r_i)$ to each party P_i.
2. P_i generates a signature σ_i on $(\mathsf{com}_i, \mathsf{sid})$, and sends σ_i to P_{n+1}.
3. If any P_i fails to send a valid signature, then P_{n+1} sends (independent) dummy commitments to all parties. Otherwise, P_{n+1} sends an independent commitment to $(\mathsf{com}_1, \sigma_1, \ldots, \mathsf{com}_n, \sigma_n)$ to all parties. In either case, P_{n+1} then proves to each party in zero knowledge that the commitments it sent takes one of these forms.

The actual protocol $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$ is slightly more complex. Furthermore, for technical reasons we do not use commitments, signatures, or zero-knowledge proofs directly but instead work in the ($\mathcal{F}_{\mathsf{com}}$, $\mathcal{F}_{\mathsf{Sign}}$, $\mathcal{F}_{\mathsf{zk}}$)-hybrid model. The complete protocol and a proof of security are given in the full version of the paper.

3.4 A Protocol Π for Collusion-Free Secure Computation

We now describe a collusion-free protocol Π that securely computes $\mathcal{F}$ in the $(\mathcal{F}_{\mathsf{com}}, \mathcal{F}_{\mathsf{ct}}, \mathcal{F}^{\pi}_{\mathsf{compute}}, \mathcal{F}^{\pi}_{\mathsf{output}}, \mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}})$-hybrid model. When these functionalities are realized using protocols designed for the mediated model, we obtain a protocol for the real mediated model.

Our protocol consists of three stages. In the first stage, the parties commit to their inputs and random coins for a protocol π that securely computes $\mathcal{F}$ (in the standard sense). In the second stage, the parties simulate π, round-by-round, as follows. If it is P_j's turn to broadcast (for $j \in [n]$), then P_j runs $\mathcal{F}^{\pi}_{\mathsf{compute}}$ with the mediator; thus, the mediator obtains the next π-message msg along with a signature of P_j on this message (and the current round number). If it is the mediator's turn to broadcast, the mediator simply computes the next π-message msg on its own, and then runs "mediator broadcast" using msg. As long as everyone behaves honestly, each party thus learns commitments to all messages of the protocol. In the third stage, the mediator runs $\mathcal{F}^{\pi}_{\mathsf{output}}$ with each P_j to enable P_j to learn its output. We now describe the protocol formally.

The protocol begins with each party P_i ($i \in [n]$) holding a vector KeyVec_i, and with the mediator holding $\{\mathsf{KeyVec}^i_{n+1}\}_{i \in [n]}$. Party P_i also holds input x_i and, if $i \in [n]$, its own secret key sk_i.

Stage 1 – input commitment and coin tossing:

1. Each P_j executes $\mathcal{F}_{\mathsf{com}}$ with P_{n+1}, where P_j chooses random s_j and provides input $\mathsf{dec}^{\mathsf{input}}_j = (x_j, s_j)$ to $\mathcal{F}_{\mathsf{com}}$. Let $\mathsf{com}^{\mathsf{input}}_j$ be the commitment received by P_{n+1} from $\mathcal{F}_{\mathsf{com}}$.
2. Each P_j executes $\mathcal{F}_{\mathsf{ct}}$ with P_{n+1}, where the input length ℓ is the number of coins needed to run π. We denote by $\mathsf{dec}^{\mathsf{rand}}_j = (r_j, s'_j)$ the output of P_j and by $\mathsf{com}^{\mathsf{rand}}_j$ the output of P_{n+1}.

Stage 2 – round-by-round emulation of π: The mediator P_{n+1} initializes $\mathsf{abort} = \mathsf{false}$. Then, for $i = 1$ to $r - 1$, the parties run the following:

1. **(P_{n+1} learns the round-i message of π.)**
Case 1: Party P_j, for $1 \le j \le n$, is supposed to broadcast in the ith round of π.

- Let $\boldsymbol{C}_j = (\mathsf{com}^j_1, \ldots, \mathsf{com}^j_{i-1})$ be the commitments that P_j output in the previous $i-1$ rounds.
- P_j and P_{n+1} run an instance of $\mathcal{F}^{\pi}_{\mathsf{compute}}$. Here, P_j sends $\mathcal{F}^{\pi}_{\mathsf{compute}}$ its commitments $\mathsf{com}^{\mathsf{input}}_j$ and $\mathsf{com}^{\mathsf{rand}}_j$, the vector of commitments $\boldsymbol{C}_j$, vector KeyVec_j, the round identifier $\mathsf{rid} = 0i$, the decommitments $\mathsf{dec}^{\mathsf{input}}_j$ and $\mathsf{dec}^{\mathsf{rand}}_j$, and its signing key sk_j.

  P_{n+1} sends $\mathcal{F}^{\pi}_{\mathsf{compute}}$ the commitments $\mathsf{com}^{\mathsf{input}}_j$, $\mathsf{com}^{\mathsf{rand}}_j$, and $(\mathsf{com}^j_1$, $\ldots$, $\mathsf{com}^j_{i-1})$; vector KeyVec^j_{n+1}; the round identifier $\mathsf{rid} = 0i$; and the decommitments $(\mathsf{dec}^j_1$, $\ldots$, $\mathsf{dec}^j_{i-1})$.
- If $\mathcal{F}^{\pi}_{\mathsf{compute}}$ returns $\perp$ to P_{n+1}, then P_{n+1} sets $\mathsf{abort} = \mathsf{true}$ and $m_i = \mathsf{dummy}$. Otherwise, if $\mathcal{F}^{\pi}_{\mathsf{compute}}$ returns $(\mathsf{msg}_i, \sigma_i)$ to P_{n+1}, then P_{n+1} sets $m_i = (\mathsf{msg}_i, \sigma_i)$.

Case 2: P_{n+1} is supposed to broadcast in the ith round of π:

- If $\mathsf{abort} = \mathsf{true}$ then P_{n+1} sets $m_i = \mathsf{dummy}$. If $\mathsf{abort} = \mathsf{false}$ then P_{n+1} locally computes the message msg_i as instructed by π (this is possible since P_{n+1} sees all π-messages "in the clear"), and sets $m_i = \mathsf{msg}_i$.

2. **(P_{n+1} "broadcasts" the round-i message of π.)** Let $\mathsf{sid} = 1i$. P_{n+1} chooses random $r_1, \ldots, r_n$ and runs $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$ with the other parties, where P_{n+1} provides input $(m_i, r_1, \ldots, r_n, \mathcal{H} = [n], \{\mathsf{KeyVec}^i_{n+1}\}_{i \in [n]})$ and every other party P_j provides input 1 and KeyVec_j.

   Each party P_j defines com^j_i to be the commitment that it received from $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$. Note that P_{n+1}, given its output from $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$, can compute the commitments $\{\mathsf{com}^j_i\}_{j \in [n]}$, and knows the corresponding decommitments.

Stage 3 – output determination: (Note that this emulates the rth and final round of π.)

1. If $\mathsf{abort} = \mathsf{true}$ then P_{n+1} sets $\mathsf{msg}_r = \mathsf{dummy}$ and sets $\text{OUT}_{n+1} = \perp$. If $\mathsf{abort} = \mathsf{false}$ then P_{n+1} computes its π-output OUT_{n+1} and final message msg_r locally (it can do this since P_{n+1} sees all π-messages "in the clear"). In either case, the mediator then sets $\mathsf{sid} = 1r$, chooses random $r_1, \ldots, r_n$, and runs $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$ with all the other parties, where P_{n+1} provides input $(\mathsf{msg}_r, r_1, \ldots, r_n, \mathcal{H} = [n], \{\mathsf{KeyVec}^i_{n+1}\}_{i \in [n]})$ and every other party P_i provides input 1 and KeyVec_i. The mediator outputs OUT_{n+1}.

   Each party P_j defines com^j_r to be the commitment that it received from $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$. Note that P_{n+1} can compute the commitment, and knows the corresponding decommitment.
2. The mediator P_{n+1} runs $\mathcal{F}^{\pi}_{\mathsf{output}}$ with each P_j, where P_j provides input $\mathsf{com}^{\mathsf{input}}_j$, $\mathsf{com}^{\mathsf{rand}}_j$, KeyVec_j, $(\mathsf{com}^j_1, \ldots, \mathsf{com}^j_r)$, $\mathsf{dec}^{\mathsf{input}}_j$, and $\mathsf{dec}^{\mathsf{rand}}_j$, and P_{n+1} sends $\mathsf{com}^{\mathsf{input}}_j$, $\mathsf{com}^{\mathsf{rand}}_j$, KeyVec^j_{n+1} the commitments $(\mathsf{com}^j_1, \ldots, \mathsf{com}^j_r)$, and the decommitments $(\mathsf{dec}^j_1, \ldots, \mathsf{dec}^j_{i-1})$.

   Each party P_j outputs the value it receives from $\mathcal{F}^{\pi}_{\mathsf{output}}$ in this step.

4 Proof of Security

All the results stated here apply to protocols run in either the trusted-PKI or mediated-PKI settings.

We first prove that Π is a collusion-free protocol that securely computes $\mathcal{F}$ in the ($\mathcal{F}_{\mathsf{com}}$, $\mathcal{F}_{\mathsf{ct}}$, $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$, $\mathcal{F}^{\pi}_{\mathsf{compute}}$, $\mathcal{F}^{\pi}_{\mathsf{output}}$)-hybrid model. A proof of the following appears in the full version of the paper.

Theorem 1. *Let π be a protocol that securely computes $\mathcal{F}$ (as required in Section 3.1); let C be a perfectly binding commitment scheme; and let* ($\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}$) *be a secure signature scheme. Then protocol Π from the previous section is a collusion-free protocol for securely computing $\mathcal{F}$ in the ($\mathcal{F}_{\mathsf{com}}$, $\mathcal{F}_{\mathsf{ct}}$, $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$, $\mathcal{F}^{\pi}_{\mathsf{compute}}$, $\mathcal{F}^{\pi}_{\mathsf{output}}$)-hybrid model.*

We now show that when the ideal-world functionalities are instantiated using protocols satisfying appropriate definitions of security, we obtain a collusion-free protocol that securely computes $\mathcal{F}$ in the real mediated model. We obtain this as a corollary of the following composition theorems.

Theorem 2. *Let Π be a collusion-free protocol computing $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, where Π contains polynomially many sequential calls to $\mathcal{G}$, and let ρ be a collusion-free protocol computing $\mathcal{G}$. Then the composed protocol Π^{ρ} is a collusion-free protocol computing $\mathcal{F}$ in the real mediated model.*

A proof of Theorem 2 follows along the lines of [5] and is given in the full version of the paper.

Theorem 3. *Let $\mathcal{G}$ be a two-party functionality, and let Π be a multi-party protocol that securely computes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model for concurrent self composition. Assume further that Π only makes calls to $\mathcal{G}$ (i.e., there are no other messages in Π), and that P_{n+1} plays the role of the second party in all calls to $\mathcal{G}$. Let m denote the overall number of calls to $\mathcal{G}$ in Π.*

If ρ is a two-party protocol that securely computes $\mathcal{G}$ under m-bounded concurrent self-composition, then the composed protocol Π^{ρ} securely computes $\mathcal{F}$ in the real mediated model.

Note that even if Π instructs the parties to make *sequential* calls to $\mathcal{G}$, Theorem 3 requires ρ to be secure under (bounded) concurrent self-composition since a dishonest mediator may run concurrent executions with different honest parties. Theorem 3 follows immediately from the definition of m-bounded concurrent self composition; a proof is given in the full version of the paper.

We can now prove our main result:

Corollary 1. *Let $\mathcal{F}$ be a polynomial-time, multi-party functionality. Then assuming the existence of enhanced trapdoor permutations, there exists a collusion-free protocol for securely computing $\mathcal{F}$ in the real mediated model, in either the* trusted-PKI *or* mediated-PKI *settings.*

Proof. Let Π^{cf} denote the protocol of Section 3.4, where:

- C is a perfectly binding commitment scheme and $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a secure signature scheme;
- π securely computes $\mathcal{F}$ (in the standard communication model) as specified in Section 3.1;
- $\mathcal{F}_{\mathsf{com}}$, $\mathcal{F}_{\mathsf{ct}}$, $\mathcal{F}^{\pi}_{\mathsf{compute}}$, and $\mathcal{F}^{\pi}_{\mathsf{output}}$ are instantiated by a single protocol[3] ρ that is secure under m-bounded concurrent self-composition [14,16] (m will be specified in the proof below);
- $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$ is instantiated using protocol $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$ (given in the full version of the paper) where $\mathcal{F}_{\mathsf{com}}, \mathcal{F}_{\mathsf{Sign}}$, and $\mathcal{F}_{\mathsf{zk}}$ are realized by the same protocol ρ as above.

Note that Π^{cf} is defined in the real mediated model, and all the components above can be constructed under the assumption that enhanced trapdoor permutations exist. We now prove that Π^{cf} is a collusion-free protocol securely computing $\mathcal{F}$.

In the case of an honest mediator, this follows directly from Theorems 1 and 2 using the fact that the "mediator broadcast" protocol of Section 3.3 is collusion-free and the observation that any two-party protocol secure in the standard sense is trivially collusion-free. (If ρ is secure under m-bounded concurrent self-composition, it is also secure in the stand-alone sense.)

In the case of a dishonest mediator, the proof is slightly more involved since the hybrid-world protocol Π, as specified, does not fulfill the requirements of Theorem 3 (because $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$ is not a two-party protocol). Nevertheless, observe that $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$ consists only of calls to the two-party functionalities $\mathcal{F}_{\mathsf{com}}$, $\mathcal{F}_{\mathsf{Sign}}$, and $\mathcal{F}_{\mathsf{zk}}$. Thus, if we define Π' to be the same as protocol Π but using $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$ instead of $\mathcal{F}^{\mathsf{sid}}_{\mathsf{bcast}}$, it follows that Π' *does* fulfill the requirements of Theorem 3. Observing that this changes the output distribution by at most a negligible amount (by security of $\Pi^{\mathsf{sid}}_{\mathsf{bcast}}$), we have that Π' securely computes $\mathcal{F}$ in the $(\mathcal{F}_{\mathsf{com}}, \mathcal{F}_{\mathsf{ct}}, \mathcal{F}_{\mathsf{Sign}}, \mathcal{F}_{\mathsf{zk}}, \mathcal{F}^{\pi}_{\mathsf{compute}}, \mathcal{F}^{\pi}_{\mathsf{output}})$-hybrid model. Using an appropriate protocol ρ as required by Theorem 3, where m is the total number of ideal calls in Π', we conclude that $\Pi^{\mathsf{cf}} = \Pi'^{\rho}$ securely computes $\mathcal{F}$ in the mediated model.

References

1. Alwen, J., Shelat, A., Visconti, I.: Collusion-Free Protocols in the Mediated Model. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 497–514. Springer, Heidelberg (2008)
2. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure Computation without Authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
3. Bohli, J.M., Steinwandt, R.: On Subliminal Channels in Deterministic Signature Schemes. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 182–194. Springer, Heidelberg (2005)

[3] This means we simply "wrap" these functionalities in one larger functionality, and have parties provide an additional input selecting which sub-functionality to run.

4. Burmester, M., Desmedt, Y., Itoh, T., Sakurai, K., Shizuya, H., Yung, M.: A Progress Report on Subliminal-Free Channels. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 157–168. Springer, Heidelberg (1996)
5. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13(1), 143–202 (2000)
6. Cramption, P., Schwartz, J.: Collusive Bidding: Lessons from the FCC Spectrum Auctions. Journal of Regulatory Economics 17(3), 229–252 (2000)
7. Desmedt, Y.: Simmons' Protocol is not Free of Subliminal Channels. In: IEEE Computer Security Foundations Workshop, pp. 170–175 (1996)
8. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
9. Hopper, N., Langford, J., von Ahn, L.: Provably Secure Steganography. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 77–92. Springer, Heidelberg (2002)
10. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably Secure Devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)
11. Izmalkov, S., Micali, S., Lepinski, M.: Rational Secure Computation and Ideal Mechanism Design. In: Foundations of Computer Science (FOCS) 2005, pp. 585–595 (2005)
12. Lepinski, M., Micali, S., Shelat, A.: Fair Zero-Knowledge. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 245–263. Springer, Heidelberg (2005)
13. Lepinski, M., Micali, S., Shelat, A.: Collusion-Free Protocols. In: Symposium on Theory of Computing (STOC) 2005, pp. 543–552. ACM, New York (2005)
14. Lindell, Y.: Protocols for Bounded-Concurrent Secure Two-Party Computation in the Plain Model. Chicago Journal of Theoretical Computer Science (1), 1–50 (2006)
15. Lindell, Y.: Lower Bounds and Impossibility Results for Concurrent Self Composition. Journal of Cryptology 21(2), 200–249 (2008)
16. Pass, R.: Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. In: Symposium on Theory of Computing (STOC) 2004, pp. 232–241 (2004)
17. Simmons, G.: The Prisoners' Problem and the Subliminal Channel. In: Advances in Cryptology—Crypto 1983, pp. 51–67. Springer, Heidelberg (1983)
18. Simmons, G.: Cryptanalysis and Protocol Failures. Comm. ACM 37(11), 56–65 (1994)
19. Simmons, G.: The History of Subliminal Channels. In: Information Hiding Workshop, pp. 237–256 (1996)

Privacy-Enhancing Auctions Using Rational Cryptography[⋆]

Peter Bro Miltersen[1], Jesper Buus Nielsen[1], and Nikos Triandopoulos[2]

[1] Dept. of Computer Science, Aarhus University, Denmark
[2] Dept. of Computer Science, Boston University, USA

Abstract. We consider enhancing with privacy concerns a large class of auctions, which include sealed-bid single-item auctions but also general multi-item multi-winner auctions, our assumption being that bidders primarily care about monetary payoff and secondarily worry about exposing information about their type to other players and learning information about other players' types, that is, bidders are *greedy then paranoid*. To treat privacy explicitly within the game theoretic context, we put forward a novel *hybrid utility* model that considers both monetary and privacy components in players' payoffs.

We show how to use rational cryptography to approximately implement any given *ex interim* individually strictly rational equilibrium of such an auction without a trusted mediator through a cryptographic protocol that uses only point-to-point authenticated channels between the players. By "ex interim individually strictly rational" we mean that, given its type and before making its move, each player has a strictly positive expected utility. By "approximately implement" we mean that, under cryptographic assumptions, running the protocol is a computational Nash equilibrium with a payoff profile negligibly close to the original equilibrium.

1 Introduction

1.1 The Problem: Realizing Privacy-Enhanced Auctions

Consider the following scenario: A *seller* S wants to sell some items to a subset of n *bidders* $P_1, P_2, \ldots, P_n$ using a sealed bid auction, e.g., a first-price or a second-price (Vickrey) auction if there is just one item. To optimize their expected payoff in these settings, the bidders P_i are to submit their true valuation of the items (e.g., in a Vickrey auction) or more generally a function of their true valuation (e.g., the Bayesian equilibrium strategy in a first-price auction) as their bid. However, in the scenario we suggest, matters are complicated by the following issues: First, bidders are not happy revealing any information related to their true valuation to the seller. Second, bidders would also be unhappy if other buyers gain information about their valuation. On the other hand, they would appreciate learning something about the valuations of the other players if they get the chance.

Some of these concerns can be handled by assuming the availability of a trusted *mediator* M. Such a trusted party can collect the bids, determine the winners, and ensure

[⋆] Supported by the *Center for Algorithmic Game Theory*, funded by The Carlsberg Foundation.

that the seller and the winners get in touch with one another. Ideal mediation does not solve all problems though, as the outcome potentially depends on the type of all parties. Hence a player which is paranoid enough about leaking information about its type might abstain from reporting the true valuation simply for privacy reasons. In this paper we first investigate when we can expect to find mechanisms which the parties would be willing to participate in if executed by an ideal mediator M. We then investigate how to realize such a mechanism in a world without an ideal mediator. The first problem forces us to assume that the parties are more interested in winning the good than worried about privacy. To solve the second problem we propose to replace M by a secure multiparty computation (MPC), as follows:

1. The seller commits in advance to sell the items to the bidders that can present a document digitally signed by all bidders, stating that P_i is the buyer of some given items. The document should also specify at which price P_i is to get each item.
2. The bidders perform a secure multiparty computation that simulates the mediator of the mediated auction and produces a set of such signed documents, i.e., one document per each winner associating the winner to the correct item-value pairs.

Indeed, previous papers concerned with secure cryptographic implementations of auctions have suggested schemes along these lines, e.g., [21, 18]. Also, at least in one instance such a scheme (for a double auction) has been implemented in practice [2].

There are issues that make this not quite solve our problem. As an example, the introduced privacy concerns of the bidders dictate the use of joint computations that eventually produce non-symmetric outputs for the bidders, where only the winners see their own contracts; then, nothing enforces the winners to send the contracts and complete the transaction with S. This, e.g., destroys the standard equilibrium analysis of a Vickrey auction which crucially depends on the winner being forced to buy, to make it costly to bid higher than ones valuation. This suggests using a first-price auction instead, but even then it is not obvious that rational parties with privacy concerns will carry out the protocol outlined above.

In general, we wish to extend classical equilibrium analysis of auctions of game theory to cryptographic auction protocols and make an argument that a rational party *has no incentive to deviate from following the protocol as specified.* A concrete problem is *protocol participation*. In realizations of games with non-symmetric final payoffs (like auctions), an agent has no incentive to continue and complete the protocol as soon as he realizes that he cannot be a winner. In contrast, the traditional analysis of multiparty computation assumes that at least *some* parties are "honest" and will carry out all steps of the protocol, no matter what (Bradford *et al.* [3] study the problem of protocol-completion incentives that exist in an auction when participants realize that they cannot win the auction, but in a model where privacy is not captured in players' rationality). Many works on rational cryptography have analyzed secret sharing and multiparty computation as a game [12, 11, 9, 1, 7, 16, 20] but, aiming at simultaneous information exchange and modeling rationality through pure information loss/gain, these works cannot precisely model auctions with non-symmetric outcomes/payoffs and a setting where utilities are a mix of monetary utilities and privacy concerns.

Matters are complicated by the fact that even the mediated auction *does* leak some information (e.g, the mere fact that a bidder did not win gives him information about the winning bid(s)). Hence, it is intuitively clear that if the privacy has high weight, existing equilibria in the classical case are disturbed (e.g., truth telling is no longer even a Nash equilibrium for Vickrey auctions), and for a high enough emphasis on privacy, not taking part in the auction (say, by submitting the bid 0, independently of the valuation) becomes a strictly dominant strategy. Whatever analysis one obtains will have to be consistent with this fact.

Perhaps the biggest challenge, finally, is to design a protocol as the above in a way that can be realized using *today's Internet computing and communication machinery*. While there are results that allow removing mediators in very general classes of games [10, 13, 14], these works use communication channels such as simultaneous broadcast (like most works on rational cryptography) or physical envelopes that are quite restrictive or even unobtainable when considering a practical Internet-based implementation.

1.2 Outline of Our Contribution

In this paper, we suggest a rational cryptographic protocol for replacing a trusted mediator in a large class of auctions. The protocol uses only point-to-point authenticated channels between the buyers, and can therefore be implemented on the Internet.

We propose a protocol where the seller does not participate. If we allowed the seller to be an active entity in the protocol execution some steps of the protocol could be significantly simplified, but a solution without seller participation has the potential to allow for more applications. As an example, a resource-limited device outsourcing computations might prefer the potential companies to execute the auction determining the winning company-price pair and just have them inform it of the outcome. As described above, the outcome of the protocol is determined by the winners getting contracts digitally signed by all other participants. How such a digitally signed contract is enforced is not our concern here. We simply assume that such bit strings have monetary value.

Besides such monetary concerns, we have to assign utilities to players so that the privacy concerns outlined in the previous subsection are adequately modeled. Because of the monetary value of the signed document, we deviate from previous works on secure auction implementation where privacy was treated at a second-phase technical level *outside* of the scope of game and parties' strategies, but also from previous works in rational cryptography where utilities were *solely* concerned with gain or exposure of information. Instead, we propose a *hybrid utility model* where agents are interested in both monetary gain from participating in the auction as well as in maintaining the privacy of their type (e.g., valuation). Their actual utility is a linear combination of a *monetary utility* and an *information utility*. For the information utility, rather than postulating one particular utility measure, we allow players to have *any* privacy concerns, under a few technical restrictions, like not positively valuing loss of information. We note that a different hybrid utility model is studied by Halpern and Pass [8].

We consider a general class of auctions in the standard Bayesian setup of auction theory and *without* privacy concerns. We formally define the corresponding mediated game *with* privacy concerns, as modeled using our hybrid utilities. In general, as we indicated in an intuitive way in the previous subsection, if high weight is put on the information part of the hybrid utilities, then the equilibria of the privacy-aware game may be very different from the equilibria of the original game. However, for many interesting cases of auctions, for instance in a variant of the first-price auction with discrete valuations and bids, we observe that when the weight put on the information concern is "smaller" than the weight on the monetary concern, then the original auction mechanism (with a small twist) is an equilibrium of the mediated game.

To study auctions with privacy concerns for the Internet, where the seller does not participate, we introduce *mediation with reject*, a slightly relaxed mediated setting where the winners are given the choice to reject their contracts. This captures the following issue: at some specific point in the computation, the winners (and only those) will locally compute their contracts (similar to the *revelation point* of [12]); nothing prevents them from not sending the contract to the seller. As we will see, the reject option can drastically affect the equilibria.

Our main result is the following. We can relate a given equilibrium (suggested behavior) π of the mediated game to a corresponding suggested behavior π' of our unmediated cryptographic protocol so that π' has the same payoff profile as π, up to a negligible amount, and for computationally bounded agents following the protocol π' is an ϵ-Nash equilibrium where ϵ is negligible. Here, "negligible" is defined relative to the strength of the cryptography used. The assumption we need is the following: The equilibrium π should have an *ex interim* expected monetary utility for all players which is large compared to the players' privacy concerns. That is, after a player learns his type, but before he makes his move, his expected conditional monetary utility is large compared to how concerned he is about privacy—parties are "greedy-then-paranoid".

As an example, our protocol enables a variant of the first-price auction and the corresponding Bayesian bidding equilibrium to be conducted by computationally bounded, rational but *not* necessarily honest buyers over the Internet in a realistic way, without a trusted mediator and without participation of the seller. In this regard, our results can be viewed as a more realistic step towards privacy-aware extensions of computational and distributed mechanism design (e.g., Ch. 14 of [19]).

We remark that while Kol and Naor [11] identify ϵ-Nash equilibrium as a minimum rationality requirement for rational cryptography, a body of works [9, 1, 7, 16, 11, 12, 17], suggest using stronger solution concepts, most notably *iterated admissibility*, and equilibria that are *not susceptible to backward inductions* [11]. However, at the time of writing, there is no clear consensus about which equilibrium refinement is the "right one" for rational cryptography. This is especially true for the computational setting where one must refine computational Nash (i.e., ϵ-Nash) equilibrium rather than Nash equilibrium: while there is a significant body of game theoretic literature about refining exact Nash equilibrium that one can draw upon, there is little or no help from the game theory community about refining approximate Nash equilibrium.[1] We note that Kol and Naor [12] strongly argue that iterated admissibility is not an appropriate concept to use. We want to add the following observation. Computational Nash equilibrium is a solution concept for games played by *software*, not conscious agents. Thus, when we ponder whether a given equilibrium is sufficiently stable or whether deviations will be made, it seems that we should focus on whether the software will be modified *before*

[1] There is a good reason for this: many or most standard equilibrium refinements are defined or motivated by players caring about *infinitely* small differences in payoff. This is inconsistent with the philosophy of ϵ-Nash in a fundamental way.

it is executed, e.g., at the moment when a player learns his type (i.e., *ex interim*) rather than whether deviations will take place *during play* at particular information sets. In other words, we propose the following thesis: Meaningful refinements of computational Nash equilibrium should be definable in the *normal form* of the game, rather than the *extensive form*. We note that the concerns about susceptibility to backward induction raised by Kol and Naor are in fact not consistent with the conjunction of this thesis and the basic assumption underlying ϵ-Nash: That players do not care pursuing advantages that are negligible. We expect much interesting work in the future about how to refine computational Nash appropriately, but in the meantime we take the standpoint that even ϵ-Nash is a meaningful property as a minimal requirement for stability, and in some cases, such as ours, it is not trivial to achieve even this.

Sketch of the protocol. The idea behind our protocol is intuitive and quite simple. Given individual signing keys and corresponding (publicly known) verification keys for some signature scheme, and also their private bids, the agents engage a randomized joint computation during which the winners obtain digital contracts signed by all agents. Conceptually, the protocol is divided in a fixed (and large) number E of stages, called *epochs*. Sequentially during each epoch e, each agent P_i receives a value $V_{e,i}$ and thus has the opportunity to obtain a contract. The contracts are released to the winners during one, randomly chosen epoch $e_0 \in [E]$ (with probability 2^{-e} in epoch $e = 1, \ldots, E-1$), whereas all other received values (by non-winners P_i in epoch e_0, or by any agent at all other epochs) are set to a special nil value $\perp$. This randomized functionality is implemented by first using secure multiparty computation, at the end of which each agent P_k obtains an *additive* share of each value $V_{e,i}$ (or $\perp$ if agents provide invalid inputs). From this point on, the E epochs of the protocol are realized sequentially, by simply asking in a round-robin fashion each agent to send its share of $V_{e,i}$ to P_i, and repeat for all $i = 1, \ldots, n$. Agent P_i is asked to refuse to send his shares in subsequent reconstructions, as soon as he experiences denial to reconstruct his own value $V_{e,i}$.

To see why several epochs are needed, consider a solution where the contracts are always handed out in epoch $e_0 = 1$. If P_1 does not get a contract in round 1, it knows that some other P_i is the winner, hence P_i will receive a contract in round i. This contract might contain information on P_1's type, which means that P_1 might have incentive to make the protocol abort, by not sending its share. We deal with this using the, by now, standard trick of not having a known epoch in which the outcomes are revealed, to ensure that with positive probability any agent deviating at epoch $e < E$ destroys his winning possibility in a later epoch. This does not hold in epoch E, but $e_0 = E$ occurs only with negligible probability, so the protocol is an ϵ-Nash for a negligible ϵ.

When there are several winners, the above protocol does not work: A winner P_i already having received his contract could have incentive to make the protocol abort before the other winners received their contracts, as these contracts could contain information related to P_i's type. To solve this issue we let the winners learn all the information in their contracts in epoch e_0, but in an unsigned form. Then in epoch $e_0 + 1$ we let them learn their signed contracts. Now, when P_i gets his contract, it is too late to prevent the other winners from learning the information in their contracts, and the contracts themselves contain no new information. Depriving other winners of their contracts would only change *their* monetary utility, and we do not model envy.

Inspired by early work on rational cryptography (e.g., [9, 1, 7, 16]) this epoch-based protocol design has been recently used along with sequential revealing of secrets to achieve complete fairness in joint computations and information exchanging (e.g., [6, 12]). The non-symmetric outcomes in auction games and the use of only point-to-point communication create a different setting where our protocol operates in. But what further distinguishes our work is how fairness is reached between the many "greedy-then-paranoid" winners: the *decoupling* of the revelation of their winning state from the (subsequent) release of their winning award in combination with bidders rationality can guarantee protocol termination.

Paper structure. In Section 2 we provide a brief description of the classical auctions model in the (pure) mediated setting. In Section 3 we introduce a definitional framework for protocol games. In Section 4 we present the mediated setting with reject and discuss the existence in this model of privacy-enhanced Nash equilibria for first-price auctions. In Section 5 we present our protocol for realizing auctions over the Internet. In Section 6 we introduce privacy-enhanced Nash realization, our core proof technique for designing and proving privacy-enhanced Nash equilibria in a modular manner.

2 Classical Auctions

First, we recap the classical (i.e., privacy-oblivious) model of a sealed-bid auction as a Bayesian game with incomplete information. Such a game is played by parties (bidders) $P_1, P_2, \ldots, P_n$ competing for one or more items to be sold. The game starts with each bidder P_i receiving a private *type* $t_i \in T_i$ where T_i is the *type space* of the bidder. The vector $t = (t_1, t_2, \ldots, t_n)$ is drawn at random from a commonly known distribution on $T = T_1 \times T_2 \ldots \times T_n$. This distribution is known as the *common prior* and will also be denoted by T. Based on his type, bidder P_i strategically chooses and submits a *bid* b_i. That is, a *strategy* of party P_i is given by a map B_i mapping types to bids. Based on the bids $b = (b_1, b_2, \ldots, b_n)$ and possibly a random source, an *allocation mechanism* Mec now allocates the items to bidders and for each item computes a *price*. We write $(o_1, \ldots, o_n) = \mathrm{Mec}(b)$, where o_i is the *outcome* for P_i—i.e., o_i specifies which items P_i won and at which prices. The *monetary utility* of a winner P_j is $r_j = g(t, o)$ for some function g, while the payoff of a non-winner P_i is $r_i = 0$. As an example, in a single-item auction t_j could be the valuation of the item, o_j could specify the winning price p and r_j could be $t_j - p$ (this is the case for a risk neutral agent P_j as he gets the item at price p and values it t_j). For the case of the Vickrey auction, the winner P_j is the bidder with the highest bid, while the corresponding winning price p is the highest bid if the bid of the winner is removed. A Bayes-Nash (or simply Nash for brevity) equilibrium for the auction is a (possibly randomized) bidding strategy maximizing the expected payoff of each bidder, if other bidders follow their prescribed strategy.

3 Protocol Games

To enhance the classical auction with privacy concerns, we have to explicitly model privacy as part of the utility function and consider appropriate notions of equilibria. For this we in turn have to explicitly model the communication of the protocol, and the information collected by a party during the protocol execution.

3.1 Communication and Protocol Execution

We start with a formal communication and protocol execution model. It is convenient to use a unified model, which allows to capture both the mediated setting and the Internet-like setting using the same formalism, which we will call a *communication device*. To be able to use cryptography, we also want to model the fact that parties are computationally bounded to get the desired definitions; this we do by simply restricting the strategy space to poly-time strategies. The model we present in this section is not specific to auctions.

Communication devices. A protocol is of the form $\pi = (\pi_1, \dots, \pi_n)$, where π_i is a program describing the strategy of party P_i. These programs communicate in rounds using a communication device $\mathcal{C}$. In each round, $\mathcal{C}$ takes an input $m_i \in \{0,1\}^d$ from each π_i and outputs a value $o_i \in \{0,1\}^d$ to each π_i. I.e., in each round, $\mathcal{C}$ is a function $(\{0,1\}^d)^n \to (\{0,1\}^d)^n, (m_1, \dots, m_n) \mapsto (o_1, \dots, o_n)$. Which function is computed might depend on the inputs and outputs of previous rounds and the randomness of $\mathcal{C}$.

Parties and strategies. We let the strategy π_i for each party P_i be an interactive circuit for R rounds. The circuit consists of $1+R$ circuits $\pi_i^{(0)}, \pi_i^{(1)}, \dots, \pi_i^{(R)}$. The circuit $\pi_i^{(0)}$ takes $a+b$ bits as input and outputs $a+b$ bits, where a, b are integers specified by the circuit. In each round π_i takes as input a *state* $s \in \{0,1\}^a$, and a *message* $m \in \{0,1\}^b$ (from the communication device $\mathcal{C}$). The output of the circuit is parsed as an updated state $s' \in \{0,1\}^a$ and a message $m' \in \{0,1\}^b$ (for device $\mathcal{C}$). Initially, the state consists of a uniformly random bits and the message is P_i's type. In subsequent rounds, s is the updated state s' from the previous round and m is the value sent by $\mathcal{C}$ for that round.

Because we consider protocols using cryptography, we do not consider a single circuit π_i. Rather π_i specifies a family of circuits, namely a circuit $\pi_i(\kappa)$ for each value κ of the security parameter.[2] Each $\pi_i(\kappa)$ is allowed to have different state and message lengths $a(\kappa), b(\kappa)$. Similarly we let $\mathcal{C}$ specify a communication device $\mathcal{C}(\kappa)$ for each $\kappa \in \mathbb{N}$. Also, for technical reasons we adopt a non-uniform model, where the sequence of strategies $\pi_i(1), \pi_i(2), \dots$ need not have a uniform description.[3] For a function $\tau : \mathbb{N} \to \mathbb{N}$ we use Π^τ to denote the *strategy space* consisting of all circuit families π_i where for all κ the size of $\pi_i(\kappa)$ is at most $\tau(\kappa)$. A strategy space Π^τ is always defined in context of some communication device $\mathcal{C}$ which for each κ expects (and produces) messages of some fixed size $d(\kappa) \in \mathbb{N}$. We require that Π^τ only contains circuit families where $b(\kappa) = d(\kappa)$ for all κ.

[2] The value of κ determines the key lengths of the underlying cryptographic primitives.

[3] Insisting on π_i having a uniform description might make it impossible to analyze the games for different values of κ independently, or would at least require an explicit argument that this can be done: Changing the strategies $\pi_i(\kappa)$ for some values of the security parameter κ might necessitate a change for other values to ensure that the sequence $\pi_1(1), \pi_1(2), \dots$ still has a uniform description. The utility of changing strategy for one specific game (i.e., for a fixed κ) might therefore not be possible to define without considering the utility of changing strategy at other security levels, which seems unintuitive and might unnecessarily complicate analysis. Adopting a non-uniform model deals with such concerns in a straight-forward manner.

Executions. Let $\mathcal{C}$ be some communication device, let $\pi = (\pi_1, \ldots, \pi_n)$ be a protocol, where $\pi_i \in \Pi^\tau$, and let T be a distribution on types. An *execution* proceeds as in Fig. 1. We call $o = (o_1, \ldots, o_n) = (o_1^{(R)}, \ldots, o_n^{(R)})$ the *outcome* of the protocol. I.e., the outcome is the last round of outputs from $\mathcal{C}$. We call the output $w_i = (s_i^{(R+1)}, m_i^{(R+1)})$ of the last circuit $\pi_i^{(R)}$ of strategy π_i the *local output* of party P_i, and call $w = (w_1, \ldots, w_n)$ the *local outputs*. We use $(t, o, w) \leftarrow (\pi, \mathcal{C})(T)$ to denote the distribution of (t, o, w) on a *random execution*, i.e., for uniformly random initial states ρ, random $t \leftarrow T$ and uniform randomness of $\mathcal{C}$.

1. Sample $(t_1, \ldots, t_n) \leftarrow T$ and uniformly random $\rho_i \in \{0,1\}^a$ for $i = 1, \ldots, n$.
2. For $i = 1, \ldots, n$, run $\pi_i^{(0)}$ on (ρ_i, t_i) to produce $(s_i^{(1)}, m_i^{(1)})$. Then for each round $r = 1, 2, \ldots, R$: First run $\mathcal{C}$ on $(m_1^{(r)}, \ldots, m_n^{(r)})$ to produce $(o_1^{(r)}, \ldots, o_n^{(r)})$, and then, for $i = 1, \ldots, n$, run $\pi_i^{(r)}$ on $(s_i^{(r)}, o_i^{(r)})$ to produce $(s_i^{(r+1)}, m_i^{(r+1)})$.

Fig. 1. An execution

Utilities. The *utility* of P_i is a real valued function u_i. We assume that u_i is a function of the types, the outcomes and the local outputs. We use u to denote $(u_1, \ldots, u_n)$. We use $u_i(T, \pi, \mathcal{C})$ to denote the *expected utility* of P_i, i.e., the expected value of $u_i(t, o, w)$ for a random execution $(t, o, w) \leftarrow (\pi, \mathcal{C})(T)$.

3.2 The Mediator and the Internet as Communication Devices

For analyzing protocols for Internet-like networks we need a communication device $\mathcal{C}^{\mathtt{int}}$ modeling communication on the Internet. Ideally we want $\mathcal{C}^{\mathtt{int}}$ to closely reflect how messages are delivered on the Internet. Since our results are very robust with respect to the exact specification of $\mathcal{C}^{\mathtt{int}}$ we will, however, use a rather idealized device.

A communication device $\mathcal{C}^{\mathtt{int}}_{\mathrm{gen},\mathrm{Out}}$ parametrized by gen and Out works as follows:

set up PKI: In round 1, sample a key pair $(pk_i, sk_i) \leftarrow \mathrm{gen}(1^\kappa)$ for each P_i and output $((pk_1, \ldots, pk_n), sk_j)$ to P_j for $j = 1, \ldots, n$.

protocol execution: In rounds $r = 2, \ldots, R-1$, the input from each party P_i is parsed as a message $m_i \in \{0,1\}^k$ for some fixed k. The output to $P_{r \bmod n}$ is $(m_1, \ldots, m_n)$. The output to all other parties is `silence`.

define outcome: In round $r = R$, compute $(o_1, \ldots, o_n) = \mathrm{Out}(msg)$, where msg are all messages sent in the previous rounds, and output the outcome o_i to P_i.

Fig. 2. An Internet-Like Device $\mathcal{C}^{\mathtt{int}}_{\mathrm{gen},\mathrm{Out}}$

We assume that the device can deliver secure messages directly between each pair of parties. This can be achieved using standard Internet technology, e.g., by establishing SSL connections between each pair of parties. Using such a model we avoid the introduction of unnecessary complications, like the exact structure of the network used to carry the messages. On the other hand, we do not want the simplification of $\mathcal{C}^{\mathtt{int}}$ to make the model unrealistic. One issue which we explicitly do not want $\mathcal{C}^{\mathtt{int}}$ to allow is simultaneous message exchange. We do this by saying that on $\mathcal{C}^{\mathtt{int}}$, in each round one predefined party receives messages from all other parties. Finally we assume the existence of a public-key infrastructure PKI. We model this in a simplistic manner by letting the device distribute the keys. In the last round the device will define an outcome, by the last set of messages output to the parties. We assume that this is a function Out of all the messages sent in previous rounds. Details are given in Fig. 2.

The communication device $\mathcal{C}^{\mathtt{med}}_{\mathrm{Mec}}$ for standard mediation is $\mathcal{C}^{\mathtt{rej}}_{\mathrm{Mec}}$ in Fig. 3 on page 552, but without **allow reject**. The recommend strategy $\pi^{\mathtt{med}}_j$ for each P_j is to input $b_j \leftarrow B_j(t_j)$ and to locally output $w_j = (t_j, o_j)$.

3.3 Information and Monetary Utilities

Information utilities. We now turn our attention to the valuation of the information collected and leaked during the protocol execution. For this we use the local outputs.

We let the local output w_i capture the type information collected by P_i. I.e., if P_i wants to take some type information with it from the execution, it outputs it as part of w_i. We assume that P_i valuates the type information collected using an *information utility* $q_i(t, w)$. Note that q_i can measure information collected by P_i as well as by other parties: maybe $q_i(t, w) = 1$ if $w_i = t_1$ but $q_i(t, w) = -1$ if $w_1 = t_i$, where $i \neq 1$.

We allow q_i to express *arbitrary* privacy concerns, except for two restrictions: To ensure that q_i is consistent with the view of knowledge from cryptography, where knowledge is the information which can be computed in poly-time, we require that q_i is poly-time computable. We also need that q_i does not positively valuate loss of information. Let $(w_1, \ldots, w_n)$ be any distribution and let $(w'_1, \ldots, w'_n)$ be the distribution where $w'_i = f(w_i)$ for a poly-time function f and $w'_{-i} = w_{-i}$. Then we require that $q_i(t, (w'_1, \ldots, w'_n)) \leq q_i(t, (w_1, \ldots, w_n)) + \epsilon$, where ϵ is negligible. In words: losing information about w_i (we think of $f(w_i)$ as throwing away information about w_i), and all other things being equal, cannot be valuated as significantly positive by P_i. We call q_i *admissible* if it has these two properties. Below we assume that all q_i are admissible.

Our protocols will work only for privacy concerns which are sufficiently small compared to the expected utility of playing the game. So it is convenient to have a measure of the privacy concerns: For an information utility $q_i(t, w)$ we call $\|q_i\| = \max_{t,w} q_i(t, w) - \min_{t,w} q_i(t, w)$ the *weight of the information utility* or *privacy weight*.

We will not be concerned about how the utility q_i measures privacy concerns, as we are going to develop protocols that are ϵ-Nash *for all admissible measures* $q = (q_1, \ldots, q_n)$ with sufficiently small weight compared to the expected monetary utility.

Monetary utilities. Complementing the information utility we have the notion of a *monetary utility*, which is just a utility function $r_i(t, o)$ that depends only on the types and the outcomes. For generality we allow r_i to change with κ. We do, however, assume that the absolute value of r_i is bounded by a polynomial in κ. The intuitive reason for this assumption is that we need to use cryptography, which withstands only poly-time attacks. In concrete terms, if you use a protocol where it would cost \$1000000 to buy enough computing power to break the cryptography, do not use it to play a game where anyone can win \$1000001. Bounding the monetary utility by a polynomial can be seen as an extremely crude way to deal with the price of computation in the utility function.

We design mechanisms which work only if the expected monetary utility of the parties is large compared to how they valuate information. We define a measure of this. For any t_i occurring with non-negligible probability as component i in $(t_1, \ldots, t_n) \leftarrow T$, let $(t, o, w) \leftarrow (\pi, \mathcal{C})(T)_{t_i}$ denote the conditional distribution of $(t, o, w) \leftarrow (\pi, \mathcal{C})(T)$ given that the i'th component of t is t_i, and let I_i denote the expected value of $u_i(t, o, w)$ for $(t, o, w) \leftarrow (\pi, \mathcal{C})(T)_{t_i}$. We call I_i the *ex interim* expected utility of P_i for t_i, i.e. its expected utility after seeing type t_i. For a given security level κ we let $\gamma(\kappa)$ be the minimum over all parties P_i and all t_i of the *ex interim* expected utility of P_i given t_i. We call $\gamma : \mathbb{N} \rightarrow \mathbb{R}$ the ex interim *rationality* of $(T, \pi, \mathcal{C})$.

3.4 Privacy-Enhanced Nash Equilibrium

When we design a mechanism, we can control the monetary utility $r_i(t, o, w) = r_i(t, o)$. In principle parties can have arbitrary utilities $u_i(t, o, w)$, even if running a protocol with the purpose of implementing some mechanism. However, we only consider settings where the part of the utility which cannot be explained as monetary utility from the designed mechanisms can be explained by an admissible measure of privacy. I.e., we assume that $q_i(t, o, w) = u_i(t, o, w) - r_i(t, o)$ is an admissible measure of privacy, s.t. $q_i(t, o, w) = q_i(t, w)$. Hence $u_i(t, o, w) = r_i(t, o) + q_i(t, w)$.

For the later schemes involving cryptography, we follow Kol and Naor [11] who argued that ϵ-Nash equilibrium for negligible ϵ is the appropriate minimum rationality requirement for "information games".

Definition 1. *For a single protocol π (i.e., for fixed κ), a strategy space Π^τ, a distribution T on types, and $\epsilon \in \mathbb{R}$, $\epsilon > 0$, we call π an ϵ-Nash equilibrium (for T, $\Pi^\tau, \mathcal{C}$) if it holds for all parties P_i and all $\pi_i^* \in \Pi^\tau$ that $u_i(T, (\pi_i^*, \pi_{-i}), \mathcal{C}) - u_i(T, \pi, \mathcal{C}) \le \epsilon$. For a protocol π (specified for all κ), strategy space Π^τ, a distribution T on types, we call π a* computational Nash equilibrium *(for T, $\Pi^\tau, \mathcal{C}$) if for all polynomials τ there exists a negligible ϵ such that $\pi(\kappa)$ is an $\epsilon(\kappa)$-Nash equilibrium (for T, $\Pi^{\tau(\kappa)}, \mathcal{C}$) for all κ.*

Our notion of computational Nash is technically slightly different from the original notion introduced by Dodis *et al.* [4], in that we use a non-uniform model, as motivated before. The notion is, however, similar enough that we feel that we can soundly reuse the terminology of a computational Nash equilibrium.

As already mentioned, implementations of monetary mechanisms can only be expected to work if the weight of the privacy concerns is relatively small. We thus capture the size of the information utility in the definition of privacy-enhanced Nash equilibria.

Definition 2. *Fix a monetary utility r and a privacy weight α. We call a protocol a* privacy-enhanced Nash equilibrium *(for r and α) if it is a computational Nash equilibrium for $u = r + q$ for all admissible privacy measures q with $\|q\| \triangleq \max_i \|q_i\| \le \alpha$.*

In words, a privacy-enhanced Nash equilibrium has the property that no matter how the parties valuate information (as long as it has weight at most α), there is no deviation which will allow any party to learn more valuable information, unless such a deviation would have it lose an equivalent amount of monetary utility. This implies that there is no way a party P_j can efficiently extract knowledge from its view of the protocol extra to that of its local output w_j. If there was, it could do so and output this extra knowledge, which would make some q_i prefer this. Therefore the recommended local outputs of a privacy-enhanced mechanism precisely specify what information each party can collect; not as an explicit requirement, but because we use computational Nash equilibrium as solution concept.

We extend the previously defined notions to cover also collusions of size t. In Definition 1 we consider $C \subset \{1, \ldots, n\}$ with $|C| \leq t$ and we consider deviations π^*_C consisting of π^*_i for $i \in C$. We call π *t-resilient* if $u_i(T, (\pi^*_C, \pi_{-C}), \mathcal{C}) - u_i(T, \pi, \mathcal{C}) \leq \epsilon$ for all $i \in C$. I.e., for all collusions of size t and all possible deviations, not even a single party in the collusion gets extra utility. This directly defines the notions of *t-resilient computational Nash equilibrium* and *t-resilient privacy-enhanced Nash equilibrium*.

As a concrete example of a privacy-enhanced Nash equilibrium for an auction mechanism with standard mediation, we consider a single-item sealed-bid first-price auction with three bidders and independent private valuations, each distributed uniformly in $\{1, 3\}$. The bidding space is the natural numbers, including 0. A general theory of equilibria of first-price auctions with integral valuations and bids is the topic of a recent paper by Escamocher *et al.* [5]. For the special case at hand, it is straightforward to check that the symmetric profile $\pi = (B_1, B_2, B_3)$, with $B_1 = B_2 = B_3$, $B_1(1) = 0$ and $B_1(3) = 1$, is a Nash equilibrium of the classical (privacy-oblivious) auction. The *ex interim* expected payoff of a bidder with valuation 1 is $1/12$ and the ex interim expected payoff of a bidder with valuation 3 is $7/6$; since payoffs are strictly bigger than 0, it is easy to check that for any privacy measure with sufficiently small weight, the equilibrium persists.

4 Mediation with Reject and Predictable Mechanisms

In what follows we consider a very general class of allocation mechanisms, but with some non-trivial restrictions. A first restriction we need is that if $(o_1, \ldots, o_n) = \mathrm{Mec}(b)$, then the utility of P_i is 0 if $o_i = \texttt{sorry}$, this outcome indicating that P_i got to buy no items. Instead, we call a party P_i with $o_i \neq \texttt{sorry}$ a *winner*. Our only use of sorry is to define mediation with reject below.

Towards designing a protocol that implements an auction on an Internet-like network without the participation of the seller and that is a privacy-enhanced Nash equilibrium, we first study privacy-enhanced Nash equilibria for a highly idealized setting that better fits the real-world setting. The idealized setting that we consider is called *mediation with reject*: here, the parties are allowed to reject the outcome of the auction and receive monetary utility 0 instead of the contract. Details are given in Fig. 3 on the next page.

It is easy to check that the standard truth telling equilibrium of a second-price auction is in general *not* a privacy-enhanced Nash equilibrium in the setting of mediation with reject: The fact that the winner is not forced to make the transaction makes bidding infinity (or the highest possible bid) a dominant strategy. For non-trivial privacy concerns, this dominant strategy is also a strictly better response than truth telling to a strategy profile where the other bidders bid truthfully. Thus, mediation with reject is a setting

Parameterized by a number of rounds R, the communication device $\mathcal{C}^{\mathtt{rej}}_{\mathrm{Mec}}$ works as follows:

compute result: In round 1, take input b_i from each P_i, let $b = (b_1, \ldots, b_n)$, sample $(o_1, \ldots, o_n) \leftarrow \mathrm{Mec}(b)$, where $o_i \neq \mathtt{sorry}$ iff P_i is a winner.
allow reject: For $i = 1, \ldots, n$: Output o_i to P_i. If P_i with $o_i \neq \mathtt{sorry}$ does not input $\mathtt{accept}$ before round R, set $o_i \leftarrow \mathtt{sorry}$.
define outcome: In the last round $r = R$, output the current value of o_i to each P_i.
side-channel: In rounds $r = 2, \ldots, R-1$, allow point-to-point communication as in $\mathcal{C}^{\mathtt{int}}$.

The recommend strategy $\pi^{\mathtt{rej}}_j$ for each P_j is to input $b_j \leftarrow B_j(t_j)$ and $\mathtt{accept}$ and locally output $w_j = (t_j, o_j)$.

Fig. 3. The Mediated Setting with Reject $(\pi^{\mathtt{rej}}_B, \mathcal{C}^{\mathtt{rej}}_{\mathrm{Mec}})$ for mechanism $(B_1, \ldots, B_n, \mathrm{Mec})$

where we observe a *separation* between first-price and second-price auctions with respect to the existence of reasonable privacy-enhanced Nash equilibria, fully justifying the importance of this abstraction.

It will, however, follow from our main result that a large class of privacy-enhanced Nash equilibria for the standard mediated setting are also privacy-enhanced Nash equilibria in the mediated setting with reject. We need a definition to phrase this result.

Definition 3. *A mechanism is called* predictable *if for each P_i, each type t_i for P_i and each bid b_i for P_i the expected monetary utility of P_i,* given that P_i bids b_i and gets $o_i \neq \mathtt{sorry}$*, depends only on t_i and b_i. Furthermore, this number $m_i(t_i, b_i)$ can be computed from t_i and b_i in poly-time.*

Clearly a Vickrey auction is not predictable, as the expected utility depends on the second largest bid, but a first-price auction *is* predictable: given that a party wins, its utility only depends on its own type and bid.

We can show that if Mec is predictable and $\gamma \geq 2\alpha$ (where α is the weight of the information utility and γ is the *ex interim* rationality) and $\pi^{\mathtt{med}}_{\mathrm{Mec}}$ is a privacy-enhanced Nash equilibrium for $(T, u, \mathcal{C}^{\mathtt{med}}_{\mathrm{Mec}})$, then $\pi^{\mathtt{rej}}_{\mathrm{Mec}}$ is a privacy-enhanced Nash equilibrium for $(T, u, \mathcal{C}^{\mathtt{rej}}_{\mathrm{Mec}})$. This shows that one can construct interesting equilibria for a mediated setting with reject. The intuition why "predictable equilibria" do not have a problem with reject, follows from the proof sketch we give in Section 5.

Privacy-enhanced Nash equilibria for first-price auctions with standard mediation exist for certain settings of the parameters, as exemplified in Section 3, and these are predictable. We therefore have interesting Nash mechanisms for the mediated setting with reject. Other examples of mechanisms for which one can design mechanisms for the setting with reject include auctions where a number ℓ of uniform items are sold to bidders with unit demand, selling to the highest ℓ bidders at their bidding price—such an auction is predictable.

5 Rational Auctions for Internet-Like Networks

We now present our Internet-based and privacy-enhanced Nash-equilibrium protocol for realizing auctions.

Assigning value to signed contracts. We want an unmediated protocol for the device $\mathcal{C}^{\texttt{umed}} = \mathcal{C}^{\texttt{int}}_{\text{gen},\text{Out}}$ for gen and Out described below. For this to be meaningful we need to make explicit how the Internet protocol allocates monetary utility. This is a fundamentally problematic issue as we are, after all, considering a pure communication protocol which anyone can set up and run without money being exchanged. As indicated in the introduction, we assign monetary value to a document if it is a possible winners' outcome for Mec and is signed by all parties.

Taking uniform items, unit-demand, first-prices auctions as an example, we can make the assumption that the seller is willing to sell to the first ℓ parties presenting a document including the party's name and a price (over the reservation price), if it is signed by all parties. This immediately assigns monetary value to commonly signed contracts. One could also use society to enforce signed contracts (cf. [15]).

In more detail, we assume that the key pair generated by gen for each party P_i consists of a verification key vk_i for an existentially unforgeable digital signature scheme and the signing key sk_i. We call σ_i a *contract* on (i, o_i) if $\sigma_i = (\sigma^1, \dots, \sigma^n)$ and each σ^j is a valid signature of (i, o_i) under vk_j. We use $\text{Contract}((i, o_i), sk)$ to denote the computing of such σ_i. We define $(o_1, \dots, o_n) = \text{Out}(msg)$ by letting $o_i = O_i$ if P_i at some point sent a valid contract on (i, O_i) to itself. We let $o_j = \texttt{sorry}$ for all other parties. For a specific mechanism, we need a way to resolve what happens if a party inputs several, different signed contracts or the parties input signed contracts not consistent with an outcome of Mec. All we need for our proof to go through is that the defined outcome only depends on the contents (i, o_i) of the signed contracts and the global order in which the device received them, like for the uniform items, unit-demand, first-prices auction above.

Mediation via a secure protocol. We show how to implement a privacy-enhanced Nash $\pi^{\texttt{rej}}_{\text{Mec}}$ in the Internet setting described in the above section. The idea is to compute the outcomes $(o_1, \dots, o_n) = \text{Mec}(b)$ as in the mediated setting with reject, using a secure MPC protocol, but then release the signed outcomes in a particular manner. The release phase will consist of E so-called epochs indexed $e = 1, \dots, E$, each consisting of n tries indexed $i = 1, \dots, n$. We index a try i within an epoch e by (e, i). In try (e, i) party P_i is given a value $V_{i,e}$, if the other parties allow it. The recommended strategy is to allow all deliveries, but as soon as a party has been denied a delivery, it will deny all parties their deliveries in all following tries. There is a special epoch $e_0 \in \{1, \dots, E-1\}$. The epoch e_0 is chosen using a probabilistic function $e_0 \leftarrow \text{Epoch}(E)$, where $e_0 \in \{1, \dots, E-1\}$ and $\Pr[e_0 = e] = 2^{-e}$ for $e = 1, \dots, E-2$. If P_i is not a winner, then $V_{e,i} = \texttt{sorry}$ for all epochs e. If P_i is a winner, then $V_{e,i} = \texttt{sorry}$ for $e \notin \{e_0, e_0+1\}$, and $V_{e_0,i} = o_i$ and $V_{e_0+1,i} = \text{Contract}((i, o_i), sk)$. When P_i receives $\text{Contract}((i, o_i), sk)$, it sends it to the seller (formally it sends it to itself and the device defines P_i to be a winner, by letting o_i be P_i's final output).

We use some notation for the $V_{e,i}$ values: For any $((o_1, \sigma_1), \dots, (o_n, \sigma_n))$ and epoch $e_0 \in \{1, \dots, E-1\}$ we define $V = (V_{1,1}, \dots, V_{1,n}, V_{2,1}, \dots, V_{E,n}) = \text{Values}(((o_1, \sigma_1), \dots, (o_n, \sigma_n)), e_o, E)$, where for all P_i, $V_{e_0,i} = o_i$, $V_{e_0+1,i} = \sigma_i$ and $V_{e,i} = \texttt{sorry}$ for $e \notin \{e_0, e_0+1\}$.

We use a secure MPC to compute sharings of the values $V_{e,i}$. Given inputs $(b_1, \dots, b_n)$, the protocol securely samples $V = (V_{1,1}, \dots, V_{1,n}, V_{2,1}, \dots, V_{E,n})$ and generates sharings $(S_{1,1},\ldots,S_{E,n}) \leftarrow \text{Sharings}(V)$, where $S_{e,i} = (S_{e,i}^{(1)},\ldots,S_{e,i}^{(n)})$ is an n-out-of-n sharing of $V_{e,i}$, where the shares are authenticated such that P_i can validate their correctness. Then the protocol gives all $S_{e,i}^{(j)}$ to P_j. The MPC protocol is chosen to tolerate the active corruption of up to $t = n-1$ parties. With this threshold termination cannot be guaranteed. The protocol should, however, guarantee that all parties P_j which received an output $y_j \neq \perp$, where $\perp$ is some designated error symbol, received a correct output. Furthermore, the protocol should guarantee that $y_j \neq \perp$ for all parties if all parties followed the protocol. After the secure MPC protocol terminates, the parties reconstruct the sharings. The details of the complete protocol $\pi_{\text{Mec}}^{\text{umed}}$ are given in Fig. 4.

The unmediated protocol for communication device $\mathcal{C}^{\text{umed}}$. The recommend strategy π_j^{umed} for P_j is as follows:

1. Receive (pk, sk_j) from the communication device.
2. In the rounds with point-to-point communication, run the code of P_j in a secure MPC for the following probabilistic function f:
   - Each P_i inputs some b_i and some (pk', sk_i'), and receives output y_i, computed as:
     - If all P_i input the same pk', and sk_i' is a signature key for pk_i', then sample $(o_1,\ldots,o_n) \leftarrow \text{Mec}(b)$ and $e_0 \leftarrow \text{Epoch}(E)$. If $o_i \neq \texttt{sorry}$, then let $\sigma_i = \text{Contract}((i,o_i), sk')$. If $o_i = \texttt{sorry}$, then let $\sigma_i = \texttt{sorry}$. Let $V = (V_{1,1},\ldots,V_{E,n}) \leftarrow \text{Values}(((o_1,\sigma_1),\ldots,(o_n,\sigma_n)), e_0, E)$, sample $(S_{1,1},\ldots,S_{E,n}) \leftarrow \text{Sharings}(V)$, and let $y_i = (S_{1,1}^{(i)},\ldots,S_{E,n}^{(i)})$.
     - Otherwise, let all $y_i = \perp$.

   Use inputs $b_j \leftarrow B_j(t_j)$ and $(pk', sk_j') = (pk, sk_j)$ to the MPC.
3. Afterward, initialize a variable $d_j \in \{\texttt{allegiance}, \texttt{defection}\}$, where $d_j = \texttt{defection}$ iff the secure MPC protocol outputs $y_j = \perp$. If $d_j \neq \texttt{defection}$, then parse y_j as shares $(S_{1,1}^{(j)},\ldots,S_{E,n}^{(j)})$.
4. Use En rounds of point-to-point communication to sequentially run E *epochs*, each consisting of *tries* $i = 1,\ldots,n$. In epoch e, try i send $s_j = S_{e,i}^{(j)}$ to P_i if $d_j = \texttt{allegiance}$ and send $s_j = \perp$ to P_i otherwise. In epoch e, try j, let $(s_1,\ldots,s_n)$ be the shares just sent by $P_1,\ldots,P_n$. If any share is invalid, then let $V_{e,j} = \perp$ and $d_j = \texttt{defection}$. Otherwise, let $V_{e,j}$ be the value reconstructed from $(s_1,\ldots,s_n)$. If $V_{e,j}$ is a valid contract, then input it to $\mathcal{C}^{\text{umed}}$.
5. If in some round $V_{e,j} = o_j$ was reconstructed, then give the local output $w_j = (t_j, o_j)$. Otherwise, give the local output $w_j = (t_j, \texttt{sorry})$.

Fig. 4. The Unmediated Protocol $\pi_{\text{Mec}}^{\text{umed}}$

Theorem 1. *Let* Mec *be any predictable mechanism. Assume that* $(\pi_{\text{Mec}}^{med}, \mathcal{C}_{\text{Mec}}^{med})$ *is a privacy-enhanced Nash equilibrium, let* γ *be the* ex interim *rationality and let* α *be the weight of the information utility. If* $\gamma \geq 2\alpha$*, then* $(\pi_{\text{Mec}}^{umed}, \mathcal{C}^{umed})$ *is a privacy-enhanced Nash equilibrium with a utility profile negligibly close to that of* $(\pi_{\text{Mec}}^{med}, \mathcal{C}_{\text{Mec}}^{med})$.

Proof. (Sketch.) We want to argue that no P_i has an incentive to deviate. We look at two cases: Case I is the situation where P_i saw a reconstructed value of the form

$V_{e,i} \neq \texttt{sorry}$. Case II is the situation where a party P_i only saw reconstructed values of the form $V_{e,i} = \texttt{sorry}$.

We first argue that a party P_i in Case I has no incentive to deviate. We look at two sub-cases. First, assume that P_i received $V_{e,i} = \text{Contract}((i, o_i), sk)$. Then it can no longer gain monetary utility: it has its contract and cannot receive another one, except by breaking the signature scheme (infeasible by assumption). It cannot gain information utility either, as all information has already been handed out: When P_i has received $V_{e,i} = \text{Contract}((i, o_i), sk)$ the game is already in epoch $e_0 + 1$, and all winners P_j received o_j in epoch e_0 and $\text{Contract}((j, o_j), sk)$ leaks no information on the types extra to o_j.[4] Second, assume that P_i received $V_{e,i} = o_i$ but did *not* yet receive $\text{Contract}((i, o_i), sk)$. If P_i sends an incorrect share to any P_j, then P_j will punish back and P_i will not receive $\text{Contract}((i, o_i), sk)$. It can essentially be argued that for any deviation there is a better deviation which never inputs a bid which will lead to a monetary utility less than $\gamma/2$ if the bid wins.[5] So, we can assume that the loss of the contract gives a loss of $\gamma/2 \geq \alpha$ in monetary utility. Aborting the protocol might gain information utility by withholding some (j, o_j), but at most utility α. So by sending an incorrect share, P_i gains utility at most $\alpha - \gamma/2 \leq 0$.

We then look at a party P_i in case II and, say, in epoch e, try j. Let S be the event that all values reconstructed by P_i until now were $\texttt{sorry}$, R the event that all values o_j with $o_j \neq \texttt{sorry}$ have been reconstructed at the corresponding winners P_j, W the event that P_i is a winner, $s = \Pr[S]$, and $w = \Pr[W]$.

We consider a party P_i which only saw $\texttt{sorry}$, which means that in the view of P_i, it is a winner with probability $\Pr[W|S] = \Pr[W \wedge S]/s$, and in the view of P_i the probability that all o_j with $o_j \neq \texttt{sorry}$ have not been reconstructed is $\Pr[\bar{R}|S] = \Pr[\bar{R} \wedge S]/s$. If P_i makes the protocol abort and P_i is a winner he loses γ' in utility, where γ' is the expected utility of P_i given that he is a winner. If P_i makes the protocol abort and $\bar{R}$, then he withholds the information o_j from at least one winner P_j and therefore gains up to α in privacy utility—if R, then no information is withheld and no privacy utility is gained. Therefore the maximal gain in utility is upper bounded by $-(\Pr[W \wedge S]/s)\gamma' + (\Pr[\bar{R} \wedge S]/s)\alpha$. To show that this is non-positive it is sufficient to show that $\Pr[\bar{R} \wedge S]\alpha - \Pr[W \wedge S]\gamma' \leq 0$. We have that $\Pr[W \wedge S] = \Pr[W \wedge (e_0 > e \vee (e = e_0 \wedge i > j))] \geq \Pr[W \wedge e_0 > e] = w2^{-e}$ and $\Pr[\bar{R} \wedge S] \leq \Pr[\bar{R}] \leq \Pr[e_0 \geq e] = 2^{-e+1}$. Since γ' is the expected monetary utility when P_i is a winner, it follows that $\gamma = w\gamma' + (1 - w)0$ and $\gamma' = \gamma/w$. So, $\Pr[\bar{R} \wedge S]\alpha - \Pr[W \wedge S]\gamma' \leq 2^{-e+1}\alpha - (w2^{-e})\gamma/w = 2^{-e}(2\alpha - \gamma) \leq 0$, as $\gamma \geq 2\alpha$.

6 Nash Implementation and Hybrid Proofs

The full proof of Theorem 1 is extensive, as handling the use of cryptography posses some challenges when fleshing out the above proof sketch. We do, however, have space to describe the general proof strategy.

[4] For this argument to work it is essential that all o_i are handled out *before* the contracts σ_i: if P_i received σ_i before a winner P_j with $j > i$ received the information o_j, P_i could find utility in aborting the protocol, thus withholding the information o_j from P_j.

[5] The full argument is slightly different: The argument uses the predictability to avoid playing such bad bids, replacing them by the recommended bid—which gains utility.

The idea is to start with an idealized version of the protocol, for a device much like the mediated setting with reject, and then introduce more and more of the details and cryptographic tools, and for each step prove that the new protocol is equivalent to the previous one. The value of such an approach when using cryptographic primitives is testified by the widespread use of hybrid proofs in the cryptographic literature.

We introduce a notion of *Nash realization* which allows to structure such proofs. Consider an idealized communication device $\mathcal{C}^{\mathtt{ide}}$ (as e.g. $\mathcal{C}^{\mathtt{rej}}_{\mathrm{Mec}}$) and a recommended protocol $\pi^{\mathtt{ide}}$ for $\mathcal{C}^{\mathtt{ide}}$, as well as a closer to real-life communication device $\mathcal{C}^{\mathtt{imp}}$ (like $\mathcal{C}^{\mathtt{umed}}$) and a protocol $\pi^{\mathtt{imp}}$ for $\mathcal{C}^{\mathtt{imp}}$. We call $(\mathcal{C}^{\mathtt{imp}}, \pi^{\mathtt{imp}})$ a realization of $(\mathcal{C}^{\mathtt{ide}}, \pi^{\mathtt{ide}})$ if the parties do not have more incentives to deviate when they interact in $(\mathcal{C}^{\mathtt{imp}}, \pi^{\mathtt{imp}})$ than when they interact in $(\mathcal{C}^{\mathtt{ide}}, \pi^{\mathtt{ide}})$.

Definition 4. *Fix a distribution T on types and a monetary utility $r = (r_1, \ldots, r_n)$. Let $(\mathcal{C}^{imp}, \pi^{imp})$ and $(\mathcal{C}^{ide}, \pi^{ide})$ be two settings. We say that $(\mathcal{C}^{imp}, \pi^{imp})$ is a t-*resilient privacy-enhanced Nash realization *of $(\mathcal{C}^{ide}, \pi^{ide})$ if for all $u = r + q$, where $q = (q_1, \ldots, q_n)$ are admissible measures of privacy with weight at most α, there exists a negligible ϵ such that:*

No less utility: *For all P_l, $u_l(T, \pi^{imp}, \mathcal{C}^{imp}) \geq u_l(T, \pi^{ide}, \mathcal{C}^{ide}) - \epsilon$.*

No more incentive to deviate: *For all $C \subset \{1, \ldots, n\}$, $|C| \leq t$, all strategies $\pi_C^{imp^*}$ for $\mathcal{C}^{imp}$, there exists a strategy $\pi_C^{ide^*}$ for $\mathcal{C}^{ide}$ so that $u_l(T, (\pi_C^{ide^*}, \pi_{-C}^{ide}), \mathcal{C}^{ide}) \geq u_l(T, (\pi_C^{imp^*}, \pi_{-C}^{imp}), \mathcal{C}^{imp}) - \epsilon$ for all $l \in C$.*

Theorem 2. *For fixed T and r, it holds for all settings $(\mathcal{C}, \pi)$, $(\mathcal{D}, \gamma)$ and $(\mathcal{E}, \delta)$ that:*

Preservation: *If $(\mathcal{C}, \pi)$ is a t-resilient privacy-enhanced Nash realization of $(\mathcal{D}, \gamma)$ and γ is a t-resilient privacy-enhanced Nash equilibrium for $\mathcal{D}$, then π is a t-resilient privacy-enhanced Nash equilibrium for $\mathcal{C}$ with a utility profile negligibly close to that of $(\mathcal{C}, \gamma)$, i.e., $|u_l(T, \pi, \mathcal{C}) - u_l(T, \gamma, \mathcal{D})|$ is negligible for all P_l and for all considered $u = r + q$.*

Transitivity: *If $(\mathcal{C}, \pi)$ and $(\mathcal{D}, \gamma)$ are t-resilient privacy-enhanced Nash realizations of $(\mathcal{D}, \gamma)$ and $(\mathcal{E}, \delta)$ respectively, then $(\mathcal{C}, \pi)$ is a t-resilient privacy-enhanced Nash realization of $(\mathcal{E}, \delta)$.*

Though this theorem is fairly easy to verify, we find the notion of Nash realization an interesting conceptual contribution, as it allows to structure hybrid proofs in a game theoretic setting. The notion can also be used for other purposes. We can, e.g., show that our protocol in Fig. 4 is an $(n-1)$-resilient privacy-enhanced Nash realization of an information theoretic secure version of the protocol, where the $V_{e,i}$ values are computed by the device and leaked in the same epoch/try structure as in Fig. 4, depending on whether or not parties input `send` or `hold` in each try. Here the notion is used to analyze a property we could not have seen by only looking at equilibria in the unmediated protocol: The result shows that our use of cryptography does not give any further incentives for deviations, to *any* size of collusion, over what is present in this highly idealized setting, which gives an extra reassurance that the cryptography was used soundly.

We complete the proof by showing that the information theoretic idealization is a privacy-enhanced Nash equilibrium. By *preservation* this result carries over to the unmediated setting. In fact, designing any t-resistant privacy-enhanced Nash equilibrium for the information theoretic setting would directly give one for the Internet too.

References

[1] Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: PODC 2006, pp. 53–62. ACM, New York (2006)

[2] Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)

[3] Bradford, P.G., Park, S., Rothkopf, M.H., Park, H.: Protocol completion incentive problems in cryptographic Vickrey auctions. Electronic Commerce Research 8(1-2), 57–77 (2008)

[4] Dodis, Y., Halevi, S., Rabin, T.: A cryptographic solution to a game theoretic problem. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 112–130. Springer, Heidelberg (2000)

[5] Escamocher, G., Miltersen, P.B., Santillan-Rodriguez, R.: Existence and computation of equilibria of first-price auctions with integral valuations and bids. In: AAMAS 2009 (2009)

[6] Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC 2008, pp. 413–422. ACM, New York (2008)

[7] Gordon, S.D., Katz, J.: Rational secret sharing, revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)

[8] Halpern, J., Pass, R.: Game theory with costly computation (manuscript) (2008)

[9] Halpern, J., Teague, V.: Rational secret sharing and multiparty computation: extended abstract. In: STOC 2004, pp. 623–632. ACM, New York (2004)

[10] Izmalkov, S., Lepinski, M., Micali, S.: Rational secure computation and ideal mechanism design. In: FOCS 2005, pp. 585–594. IEEE, Los Alamitos (2005)

[11] Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg (2008)

[12] Kol, G., Naor, M.: Games for exchanging information. In: STOC 2008, pp. 423–432. ACM, New York (2008)

[13] Lepinksi, M., Micali, S., shelat, a.: Collusion-free protocols. In: STOC 2005, pp. 543–552. ACM, New York (2005)

[14] Lepinski, M., Micali, S., Peikert, C., shelat, a.: Completely fair SFE and coalition-safe cheap talk. In: PODC 2004, pp. 1–10. ACM, New York (2004)

[15] Lindell, A.Y.: Legally-enforceable fairness in secure two-party computation. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 121–137. Springer, Heidelberg (2008)

[16] Lysyanskaya, A., Triandopoulos, N.: Rationality and adversarial behavior in multi-party computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 180–197. Springer, Heidelberg (2006)

[17] Micali, S., shelat, a.: Purely rational secret sharing (extended abstract). In: TCC 2009. LNCS, vol. 5444, pp. 54–71. Springer, Heidelberg (2009)

[18] Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: 1st International Conference on Electronic Commerce, pp. 129–139. ACM, New York (1999)

[19] Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)

[20] Ong, S.J., Parkes, D., Rosen, A., Vadhan, S.: Fairness with an honest minority and a rational majority. In: TCC 2009. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)

[21] Parkes, D.C., Rabin, M.O., Shieber, S.M., Thorpe, C.A.: Practical secrecy-preserving, verifiably correct and trustworthy auctions. In: 8th International Conference on Electronic Commerce, pp. 70–81. ACM, New York (2006)

Utility Dependence in Correct and Fair Rational Secret Sharing[⋆]

Gilad Asharov and Yehuda Lindell

Department of Computer Science
Bar-Ilan University, Israel
gilad_asharov@yahoo.com, lindell@cs.biu.ac.il

Abstract. The problem of carrying out cryptographic computations when the participating parties are *rational* in a game-theoretic sense has recently gained much attention. One problem that has been studied considerably is that of rational secret sharing. In this setting, the aim is to construct a mechanism (protocol) so that parties behaving rationally have incentive to cooperate and provide their shares in the reconstruction phase, even if each party prefers to be the only one to learn the secret.

Although this question was only recently asked by Halpern and Teague (STOC 2004), a number of works with beautiful ideas have been presented to solve this problem. However, they all have the property that the protocols constructed need to know the actual utility values of the parties (or at least a bound on them). This assumption is very problematic because the utilities of parties are not public knowledge. We ask whether this *dependence on the actual utility values* is really necessary and prove that in the basic setting, rational secret sharing cannot be achieved without it. On the positive side, we show that by somewhat relaxing the standard assumptions on the utility functions, it is possible to achieve utility independence. In addition to the above, observe that the known protocols for rational secret sharing that do not assume simultaneous channels all suffer from the problem that one of the parties can cause the others to output an incorrect value. (This problem arises when a party gains higher utility by having another output an incorrect value than by learning the secret itself; we argue that such a scenario is not at all unlikely.) We show that this problem is inherent in the non-simultaneous channels model, unless the actual values of the parties' utilities from this attack is known, in which case it is possible to prevent this from happening.

1 Introduction

Recently, there has been much interest in the intersection between cryptography and game theory [6,5,10,3,1,9,10]. One specific question that has gained much attention is that of *rational secret sharing.* The basic problem that arises when considering secret sharing (or to be more exact, protocols for the reconstruction

[⋆] Research supported by THE ISRAEL SCIENCE FOUNDATION (grant No. 781/07).

phase) is that the parties actually have no incentive to reveal their share. Specifically, assume that t parties get together to reconstruct a secret that was shared using a t-out-of-n secret sharing scheme. The way that this reconstruction takes place is simply for each party to broadcast its share to all others. However, if one party does not broadcast its share, it can still reconstruct the secret (because it received the $t-1$ shares of all other parties and so has t shares overall), but the others cannot (because they only have $t-1$ shares). Thus, under the assumption that parties prefer to be the only one to learn the secret, the rational behavior in the above naive reconstruction procedure is for every party to remain quiet and not broadcast its share [6]. The aim of rational secret sharing is therefore to construct a mechanism so that it is in the interest of rational parties to cooperate, with the result being that all parties learn the reconstructed secret. The fact that the parties are rational essentially means that they each have a utility function assigning a value to every possible outcome of the protocol (this value represents the gain that the party achieves if the given outcome occurs). Furthermore, the parties' aim is to maximize their utility. We remark that a mechanism is considered successful if it achieves a Nash equilibrium (or one of its variants) for the strategy which instructs all parties to cooperate. Loosely speaking, this means that if any one of the parties deviates from the prescribed strategy (while others follow it), then it will not obtain a higher utility (and may even lose). Thus, it is in the interest of all parties to follow the prescribed strategy and cooperate.

In order to construct a mechanism with the above properties, certain natural assumptions are made regarding the utilities of the parties. In particular, it is assumed that a party always prefers to learn the secret than to not learn it (this is essential to assume, or else there is no reason for a party to ever participate in the reconstruction). Furthermore, it is assumed that parties prefer to learn the secret, and have some or all of the other parties not learn it (when knowledge is power, this makes a lot of sense). Although the above assumptions are very reasonable, a concern with all of the known protocols is that they don't just assume that this "learning preference" holds. Rather, they assume that the *actual utility values* of the parties (or at least bounds on them) are known to all, and the mechanism itself depends on these values. The problem with this assumption is that in reality the utility of a party may not even be known to itself, let alone to others. Furthermore, even if a party knows its own utility, it is unclear how others can learn this value (it would not necessarily be rational for a party to be honest about its utility; rather, it may gain something by providing incorrect information about its utility function). This problem stands at the center of this work, and we ask the following fundamental question:

> *Is it possible to construct a single reconstruction mechanism for rational secret sharing that achieves a Nash equilibrium for all possible values of utility functions that fulfill the aforementioned assumptions regarding learning preference?*

In addition to the above, we observe that some of the known protocols suffer from a correctness issue. Specifically, most of the positive results on this topic assumed that the parties have access to a simultaneous channel (meaning that all parties can simultaneously send messages meaning that no party can see what the others broadcast before sending its own). Since simultaneous channels are problematic to implement in practice, a recent breakthrough was made that achieved rational secret sharing in non-simultaneous channels [10]. However, the protocol of [10] (and a follow-up protocol by [7]) has the problem that one of the parties can cause the others to output an incorrect value, at the expense of not learning the secret itself. Thus, the assumption made by [10] is that since a party always prefers to learn the secret, it will never follow such a strategy. However, we do not believe that this assumption is always reasonable. Rather, there are certainly scenarios where a party can gain more by having another learn incorrect information than by learning the information itself (for example, consider the case where the use of incorrect information can result in a loss of reputation, to the potential gain of others). In any case, it would certainly be preferable to not have to assume this. Noting that this problem of correctness does not arise in any of the protocols using simultaneous channels, we ask:

> *Is it possible to construct a reconstruction mechanism for rational secret sharing that uses non-simultaneous channels and achieves Nash equilibrium even if a party's utility when another party outputs an incorrect value is higher than its utility when it learns the secret? Furthermore, is it possible to achieve this without assuming knowledge of the actual utility value?*

Our results. We focus mainly on 2-out-of-2 secret sharing. Let U_i^+ denote the utility of party P_i when it learns the secret and the other party does not. Furthermore, let U_i^f denote the utility of party P_i when the other party outputs an incorrect (false) value, even if P_i itself did not learn the output. We call a mechanism U^+-independent if it achieves Nash equilibrium for all possible (polynomial) values of (U_1^+, U_2^+) that fulfill the aforementioned learning-preference assumptions (i.e., that a party prefers to learn than not learn, and prefers to be the only one to learn). We define U^f-independence similarly. We stress that when a mechanism is U^+ or U^f-independent, it may still know the values of the other utilities (i.e., the utility when all parties learn the secret or when none learn it). We begin by proving an interesting connection between U^+-independence and *complete fairness*, and between U^f-independence and *correctness* (where fairness and correctness here are in the presence of malicious adversarial behavior that may not be rational and is aimed only to break the protocol). In Section 3, we prove the following informally stated theorem:

Theorem 1. *Any two-party mechanism that achieves U^+-independence guarantees complete fairness in the presence of malicious adversarial behavior. Furthermore, any two-party mechanism that achieves U^f-independence guarantees correctness in the presence of malicious adversarial behavior.*

Intuitively, Theorem 1 holds because if a mechanism is U^+-independent, then it must be in a party's interest to cooperate even if its U^+ utility is very high.

However, if a party's U^+ utility is high enough – but still polynomial – then it can be shown that its best strategy is to just try and break fairness (because then it gains U^+). Since, it should not be able to succeed in doing this, it follows that a malicious adversary also can only break fairness with negligible probability. The connection between U^f independence and correctness is proven in a similar way. It is possible to use Theorem 1 in order to prove that there do not exist two-party reconstruction mechanisms for rational secret sharing that are independent of U^+, by showing how to toss a fair coin given any such mechanism. (Intuitively, given such a mechanism, we construct a protocol where in the first stage multiparty computation is used to generate shares of an unbiased coin, and then the mechanism is used to fairly reveal the coin.) Using the impossibility result of Cleve [2] for coin tossing, we then conclude that such a mechanism does not exist. However, we stress that unbiased coin tossing is only impossible in the non-simultaneous channels model, and thus this would only prove the impossibility of obtaining U^+-independence in this model, and leaves open the possibility that there do exist U^+-independent mechanisms in the simultaneous channels model.

We therefore provide a direct proof, ruling out the possibility of obtaining U^+-independence even when given a simultaneous channel. That is, we prove the following:

Theorem 2. *There does not exist a two-party reconstruction mechanism for rational secret sharing that is independent of U^+ in either the simultaneous or non-simultaneous channels model.*

In order to prove this, we actually present a lower bound on the number of rounds needed for achieving fair reconstruction and show that this number is dependent on the actual utility functions of the parties (or, to be more exact, a bound on them). Thus, no mechanism can be independent of the utilities because this implies that its number of rounds is also independent. Our lower bound is proven in the simultaneous-channels model and therefore also holds for non-simultaneous channels.

Having established that U^+-independence is impossible to achieve, we ask whether the other utility values must also be known. For example, we know that U^f-independence is possible in the simultaneous-channels model, because all of the known protocols for the simultaneous-channels model (cf. [5,10]) are U^f-independent. This leaves open the question regarding U^f-independence with non-simultaneous channels. We prove that:

Theorem 3. *There does not exist a two-party reconstruction mechanism for rational secret sharing that is U^f-independent in the non-simultaneous channels model.*

The proof of this theorem uses Theorem 1 that states that a U^f-independent mechanism guarantees correctness. We then prove that in the non-simultaneous channels model, it is not possible to construct a *correct* reconstruction mechanism.

Positive results. In Section 5, we present two **positive results** as follows:

1. We present a two-party reconstruction mechanism for rational secret sharing that works in the non-simultaneous model. This mechanism uses the actual values of U^f; given the impossibility result of Theorem 3, this is inherent.
2. We present a *multiparty* reconstruction mechanism that uses simultaneous channels and is *independent of all utility values*, under a relaxation of the learning-preference assumptions. Namely, we assume that a party prefers to be the only one to learn the secret but once one other party has learned the secret it makes no difference if all learn it. In fact, it suffices to assume that even though each party prefers that as few other parties as possible learn the secret, the utility if all but 1 or all but 2 parties learn is the same (i.e., it makes no difference if *all parties* learn the secret or if *almost all parties* learn the secret).

The above results show that **(a)** correctness need not be forfeited in the model with non-simultaneous channels, and **(b)** utility independence is possible to achieve in some settings, depending on the assumptions on the utility functions.

Related work. The question of rational secret sharing was first introduced by [6]. They showed that there does not exist a mechanism with a constant number of rounds, with a Nash Equilibrium that survives iterated deletions of weakly dominated strategies. Moreover, they presented a protocol for $n \geq 3$ (that is U^+-dependent) in the simultaneous model. More protocols, dealing with other settings, were presented for the simultaneous model in [5,1,9,10], and for the non-simultaneous model in [10,7]. The basic question that we ask regarding utility independence was proposed in [6]. The first partial answer to this question was given by [1] who showed that utility independence is possible for t-out-of-n secret sharing as long as $t < n/3$. This question was also considered by [14] who gave a partial answer in their model. Among other things, we have shown that it is *not* possible for the important case of 2-out-of-2 secret sharing. The works of [13,11] can be used to obtain fair secret sharing, but assume stronger physical assumptions than a simultaneous channel. Other works have also considered a mix of rational, honest and malicious parties [16,14,1].

2 Definitions and Preliminaries

We denote by $\mathcal{S}$ an efficiently samplable distribution for choosing the secret to be shared, by SHARE the secret sharing scheme and by $(\Gamma, \boldsymbol{\sigma})$ the reconstruction mechanism. Definitions of secret sharing and Nash equilibria can be found in the full version.

Outcome and utilities. The outcome of an execution of a game Γ with some strategy profile $\boldsymbol{\sigma}$ is denoted o and consists of the output of all of the parties. In the case of 2-out-of-2 secret sharing, each party may learn or may not learn the secret, and there are therefore exactly four possible outcomes. (This ignores the issue of correctness which we introduce in this paper and discuss below.)

Each party's utility is a function of these outcomes, and there are therefore also four possible utility values for each party. The notations for the four possible outcomes, and the associated utility for each party, are described in Table 1.

Table 1. Outcome and Utility

P_1 **receives** s	P_2 **receives** s	**Outcome notation**	P_1**'s Utility**	P_2**'s Utility**
NO	NO	o^{none}	U_1^-	U_2^-
NO	YES	o_2^+	U_1^{--}	U_2^+
YES	NO	o_1^+	U_1^+	U_2^{--}
YES	YES	o^{both}	U_1	U_2

In this paper, we consider the possibility that parties may output incorrect values and introduce a utility U^f for this event (informally, a party gains U_i^f if it succeeds in having the other party output a false/incorrect value). This results in *nine* possible outcomes of the game (each party may learn the correct value, not learn, or output an incorrect value). For simplicity we will consider only the outcome where one party does not learn the secret while the other outputs an incorrect (or false) value. We denote this event by o_{-i}^{false}, where P_{-i} is the party who outputs the incorrect value. (We explicitly consider this event because this is the one that occurs naturally. Needless to say, when analyzing mechanisms all possibilities need to be taken into account.)

Assumptions on the utility functions. We assume that the utility functions of all parties are polynomial in the security parameter. Formally, a party's utility function u_i is a function of the outcome and the security parameter k. We therefore write $U_i(1^k) = u_i(1^k, o^{\text{both}})$, $U_i^+(1^k) = u_i(1^k, o_i^+)$, $U_i^-(1^k) = u_i(1^k, o^{\text{none}})$, $U_i^{--}(1^k) = u_i(1^k, o_{-i}^+)$, and $U_i^f(1^k) = u_i(1^k, o_{-i}^{\text{false}})$. As is now standard [6,5,10], we assume that each party always prefers to learn the secret than to not learn it, and that each party most prefers to be the sole party to learn the secret. We add an additional assumption being that a party prefers to have the other party output an incorrect value than not, when in both cases the first party does not learn anyway. We do not make any assumption on U_i^f beyond this. (In [10] they implicitly assume that $U_i^f < U_i$ for all parties.) For lack of a better name, we call utility functions that fulfill these assumptions "natural". Formally:

Definition 4. *Let* $\mathcal{U} = \{(U_i^+, U_i, U_i^-, U_i^{--}, U_i^f)_{i\in\{1,2\}}\}$ *be a set of utility functions for the parties. We say that* $\mathcal{U}$ *is* natural *if for every* $i \in \{1,2\}$ *and for every* $k \in \mathsf{N}$ *it holds that* $U_i^+(k) \geq U_i(k) \geq U_i^-(k) \geq U_i^{--}(k) \geq 0$ *and* $U_i^f(k) \geq U_i^-(k)$.

We remark that in all previous works, it was formally assumed that $U_i^-(k) = U_i^{--}(k)$, even though none of the protocols utilized this fact. We have not defined it in this way because we find it unsatisfactory to assume that once a party has not learned, it makes no difference to its utility if others did or did not learn. On the contrary, it can be a lot worse if a party does not learn while others do learn and so protocols should take this into account. We note that all previous protocols can be modified to work with the value U_i^{--}. We also note that our lower bounds hold even if $U_i^{-} = U_i^{--}$, and so we do not assume anything about the value U_i^{--}.

Fair secret sharing. A number of different notions have been used regarding the desired equilibrium for rational secret sharing. Our impossibility results refer to the weakest of these assumptions, which is ϵ-Nash equilibrium for a negligible function $\epsilon(\cdot)$ [10,8]. However, we also require that the number of rounds be polynomial (this is needed for our lower bounds, but we argue that this does not significantly weaken our results because a mechanism with a super-polynomial of rounds is not computationally feasible to run). The natural way to model this is as a computational Nash equilibrium [3,8] (although our results hold even if local computation by each party is unbounded). We define computationally fair reconstruction mechanisms in this light:

Definition 5. *Let $\mathcal{U}$ be a set of natural utility functions for P_1 and P_2 (as in Definition 4). We say that a mechanism $(\Gamma, \boldsymbol{\sigma})$ is a* fair reconstruction mechanism for *$\mathcal{U}$ if $\boldsymbol{\sigma}$ is a computational Nash Equilibrium and if the probability that the result is not o^{both} when both parties follow $\boldsymbol{\sigma}$ is negligible.*

3 Utility-Independent Mechanisms and Properties

3.1 Definitions

We now formalize the notion of utility independence. Loosely speaking, a mechanism is independent of a given utility function if it achieves its desired properties for *any* value of that utility for all parties.

Definition 6. *Let $\hat{U} \in \{U^+, U, U^-, U^{--}, U^f\}$ be a utility type and let $\mathcal{U}' = \{U_i^+, U_i, U_i^-, U_i^{--}, U_i^f\}_{i=1}^n \setminus \{\hat{U}_i\}_{i=1}^n$ be a set of polynomial utility functions (excluding all the $\hat{U}_i$ values). We say that the mechanism $(\Gamma, \boldsymbol{\sigma})$ is a $\hat{U}$-*independent fair reconstruction mechanism *if for all polynomial utility functions $\{\hat{U}_i\}_{i=1}^n$ for which $\mathcal{U} = \mathcal{U}' \cup \{\hat{U}_i\}_{i=1}^n$ is natural, it holds that $(\Gamma, \boldsymbol{\sigma})$ is a fair reconstruction mechanism for $\mathcal{U}$.*

Note that our definition of utility independence includes the assumption that $\mathcal{U}$ is natural. In our results, we focus on U^+ and U^f independence.

Fairness and correctness. In this section, we show that U^+ and U^f independence implies the properties of complete fairness and correctness in the presence of adversarial behavior.[1] We stress that we define these notions in an *adversarial*

[1] We consider the two-party case only because we only deal with the case of no coalitions in this paper, and in the case of no coalitions we have an honest majority and so fairness and correctness (in the presence of a malicious adversary) can be achieved. This case is therefore not interesting.

context and not in a game theoretic one. That is, we say that a protocol or mechanism is completely fair/correct if it maintains this property when one of the parties follows a worst-case strategy (meaning that it has no aim to gain utility and its aim is simply to break this property of the protocol). We remark that we will move freely between protocols in a cryptographic setting with an adversary $\mathcal{A}$ and mechanisms involving rational adversaries playing a game in order to achieve utility. This is due to the fact that a mechanism trivially defines a protocol and vice versa. We now proceed to define complete fairness and correctness. We present the definitions in a "protocol context"; their translation to the game-theoretic context is discussed below. Intuitively, a two-party reconstruction protocol is *completely fair* if whenever one party learns the secret the other party is also guaranteed to learn the secret, except with negligible probability. Likewise, a reconstruction protocol is *correct* if the honest party is guaranteed to either output the correct value (i.e., the secret that was shared) or a special abort symbol $\perp$. Although it is difficult to formalize these notions for general secure computation without resorting to a full ideal model/real model definition (since the output depends on the actual inputs used by the possibly malicious parties), in the case of secret sharing it is much simpler because the output of the protocol is well defined. In particular, the output can only be the shared secret s or an abort symbol $\perp$. We assume that any reconstruction protocol is non-trivial meaning that if both parties are honest, then they both learn the secret except with negligible probability.

We first introduce some notation. Let $\text{REAL}_{\pi,\mathcal{A},i}(\text{SHARE}(\mathcal{S}))$ denote the outcome o of an execution of the reconstruction protocol π, with the parties P_1 and P_2, an adversary $\mathcal{A}$ controlling party P_i $(i \in \{1,2\})$, and a share s that was chosen by $\mathcal{S}$ and shared as in SHARE; recall that an outcome is simply the concatenation of the outputs of all participating parties (since $\mathcal{A}$ controls P_i, we consider only the output of $\mathcal{A}$ and the honest party). Next, denote by $\text{OUTPUT}_X(\text{REAL}_{\pi,\mathcal{A},i}(\text{SHARE}(\mathcal{S}))$ the output of party X (where X may be $\mathcal{A}$ or the honest party P_{-i}). Recall that the security parameter is denoted k.

Definition 7. *Let* SHARE *be a share generation algorithm for a 2-out-of-2 secret sharing scheme, and let π be the reconstruction protocol for the scheme.*

1. *We say that π is* completely fair *if for every probabilistic polynomial-time adversary $\mathcal{A}$ that controls the party P_i there exists a negligible function $\mu(\cdot)$ such that*
$$\Pr[\text{OUTPUT}_{\mathcal{A}}(\text{REAL}_{\pi,\mathcal{A},i}(\text{SHARE}(\mathcal{S}))) = \mathcal{S}] \leq \Pr[\text{OUTPUT}_{P_{-i}}(\text{REAL}_{\pi,\mathcal{A},i}(\text{SHARE}(\mathcal{S}))) = \mathcal{S}] + \mu(k).$$
2. *We say that π is* correct *if for every probabilistic polynomial-time adversary $\mathcal{A}$ that controls the party P_i there exists a negligible function $\mu(\cdot)$ such that*
$$\Pr[\text{OUTPUT}_{P_{-i}}(\text{REAL}_{\pi,\mathcal{A},i}(\text{SHARE}(\mathcal{S}))) \notin \{\mathcal{S},\perp\}] \leq \mu(k).$$

An equivalent formulation of the above for mechanisms is obtained by requiring that the result of an execution where one party follows the prescribed strategy and the other may follow *any arbitrary alternative* strategy is fair (or correct). For example, correctness of a mechanism $(\Gamma, \boldsymbol{\sigma})$ can be formalized by saying that for *every arbitrary strategy* σ_i' followed by party P_i ($i \in \{1,2\}$) there exists a negligible function μ such that:

$$\Pr[\textsc{output}_{P_{-i}}(\textsc{real}_{\Gamma, P_i(\sigma_i'), P_{-i}(\sigma_{-i})}(\textsc{share}(\mathcal{S}))) \notin \{\perp, \mathcal{S}\}] \leq \mu(k).$$

(Observe that correctness is guaranteed only when party P_{-i} follows the prescribed strategy σ_{-i}.)

3.2 U^+-Independence vs Fairness and U^f-Independence vs Correctness

We now prove that the existence of a U^+-independent reconstruction mechanism implies the existence of a completely fair reconstruction protocol. Intuitively this holds because if complete fairness is not achieved, then there exists an adversary who can participate in the protocol induced from the mechanism and with non-negligible probability can learn the secret while the honest party does not. Given such an adversary, we can set the utility U^+ of one of the parties to be high enough so that its expected gain by following the adversarial strategy is high enough. Our proof holds for both simultaneous and non-simultaneous channels.

Proposition 8. *If there exists a U^+-independent fair reconstruction mechanism for a 2-out-of-2 secret sharing scheme (as in Definition 6), then there exists a completely fair reconstruction protocol (as in Definition 7) for the scheme.*

Proof: Let $(\Gamma, \boldsymbol{\sigma})$ be a U^+-independent fair reconstruction mechanism and let $\mathcal{U}'$ be a set of utilities specifying $\{U, U^-, U^{--}, U^f\}$ for both parties. Denote by π the protocol derived from $(\Gamma, \boldsymbol{\sigma})$ as described above. Assume by contradiction that π is not a completely fair reconstruction protocol. This implies that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that controls some party P_i ($i \in \{1,2\}$) and a polynomial $p(\cdot)$ such that for infinitely many k's:

$$\begin{aligned}&\Pr\left[\textsc{output}_{\mathcal{A}}\left(\textsc{real}_{\pi,\mathcal{A},i}\left(\textsc{share}\left(\mathcal{S}\right)\right)\right) = \mathcal{S}\right]\\ &\qquad > \Pr\left[\textsc{output}_{P_{-i}}\left(\textsc{real}_{\pi,\mathcal{A},i}\left(\textsc{share}\left(\mathcal{S}\right)\right)\right) = \mathcal{S}\right] + \frac{1}{p(k)}\end{aligned}$$

Let $\sigma^{\mathcal{A}}$ be the corresponding behavioral strategy of the adversary $\mathcal{A}$ in the game Γ. Note that the outcome of the game when party P_i plays according to $\sigma^{\mathcal{A}}$, while the other party plays according to the prescribed strategy $\boldsymbol{\sigma}$, is o_i^+ with probability $1/p(k)$.

We now define the utility function U_i^+ for party P_i by $U_i^+ \geq p(k) \cdot (U_i + 1)$. We show that for infinitely many k's, P_i's utility is greater if it follows $\sigma^{\mathcal{A}}$ than if it follows σ_i, which is a contradiction to the assumption that $\boldsymbol{\sigma}$ is a (computational) Nash equilibrium. Let $\mathcal{O}$ denote the set of all possible outcomes, and recall that $u_i(o)$ is the utility of P_i upon outcome o. We have that for infinitely many k's:

$$u_i\left(\sigma_i^{\mathcal{A}}, \sigma_{-i}\right) = \sum_{o \in \mathcal{O}} \Pr[o \mid (\sigma_i^{\mathcal{A}}, \sigma_{-i})] \cdot u_i(o)$$

$$\geq \Pr\left[o_i^+ \mid (\sigma_i^{\mathcal{A}}, \sigma_{-i})\right] \cdot U_i^+$$
$$\geq \frac{1}{p(k)} \cdot (p(k) \cdot (U_i + 1)) = U_i + 1.$$

In contrast, $u_i(\sigma_i, \sigma_{-i}) = U_i$. Thus, there exists a non negligible function ϵ' (even if U_i is negligible), such that:

$$u_i\left(\sigma_i^{\mathcal{A}}, \sigma_{-i}\right) \geq u_i(\sigma_i, \sigma_{-i}) + \epsilon'(k)$$

in contradiction to the assumption that $\boldsymbol{\sigma}$ is a computational Nash equilibrium for Γ. We therefore conclude that the protocol π induced from $(\Gamma, \boldsymbol{\sigma})$ is completely fair, as in Definition 7. ■

U^f-independence implies correctness. The following is proved analogously to Proposition 8:

Proposition 9. *If a fair reconstruction mechanism for a 2-out-of-2 secret sharing scheme is U^f-independent (as in Definition 6), then it achieve correctness (as in Definition 7).*

4 Negative Results

4.1 Impossibility for U^+-Independence

As we have mentioned, Proposition 8 can be used to prove the impossibility of obtaining U^+-independent fair reconstruction mechanisms in the non-simultaneous channels model. This is because any such mechanism can be used to toss a fair coin, in contradiction to [2]. (Specifically, secure computation can be used to generate shares of a random bit, which are then reconstructed using the mechanism. By Proposition 8, this mechanism guarantees complete fairness in the presence of malicious behavior and so neither party can bias the outcome.) Such a proof leaves open the possibility of obtaining U^+-independence in the simultaneous channels model. In this section we therefore prove a lower bound on the number of rounds that are needed in any fair reconstruction mechanism, even in the simultaneous model. As we will see, the number of rounds depends on the U^+ utilities of the parties; U^+-independence is therefore not achievable.

We prove our lower bound by considering a specific attack (or, an alternative strategy) that can be carried out on every mechanism. The attack that we consider is a premature abort. When a party aborts prematurely, it does not broadcast its message in the round that it quits, while the other party does. Therefore, intuitively, it may gain more information about the secret than the other party. The mechanism must therefore guarantee that the amount of information gained in any single round is small enough so that carrying out such an attack is not profitable and will yield a lower utility. We quantify this amount of information and define an "aborting threshold" for each party as follows:

$$\beta_1 = \frac{U_1 - U_1^{--}}{U_1^+ - U_1^{--}} \qquad \text{and} \qquad \beta_2 = \frac{U_2 - U_2^{--}}{U_2^+ - U_2^{--}}$$

We now prove that the number of rounds in any fair reconstruction mechanism depends on $\{\beta_1, \beta_2\}$ and so depends on the actual utilities.

Theorem 10. *Let $(\Gamma, \boldsymbol{\sigma})$ be a fair reconstruction mechanism, let $R^{\Gamma}_{(\sigma_1,\sigma_2)}$ be a random variable denoting the number of rounds in Γ when both parties play according to $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$, and let $\beta \leq \min\{\beta_1, \beta_2\}$ be as above. Then:*

$$E[R^{\Gamma}_{(\sigma_1,\sigma_2)}] > \frac{1}{8\sqrt{\beta}}$$

Proof Sketch: We start with some notation. Denote by a_i the output of party P_1 when P_2 quits at round i *before* sending its message (that is, at round i only P_1 broadcast its message); likewise b_i denotes the output of P_2 when P_1 quits at round i. Note that when P_1 quits at round i (before sending its message) and P_2 does not quit in that round, party P_1 receives an additional message and therefore may gain additional knowledge about the secret. In such a case, P_1 outputs a_{i+1}, while P_2 outputs b_i. In the following claim, we bound the amount of additional knowledge that a party can gain in such a situation:

Claim 11. *Let $\mathcal{U}$ be a set of natural utility functions for P_1 and P_2 (as in Definition 4), and let the mechanism $(\Gamma, \boldsymbol{\sigma})$ be a* fair reconstruction mechanism for $\mathcal{U}$ *(as in Definition 5). For every round $i \geq 0$, the following must hold:*

$$\Pr[a_{i+1} = s] \leq \Pr[b_i = s] + 2\beta_1 \quad \text{and} \quad \Pr[b_{i+1} = s] \leq \Pr[a_i = s] + 2\beta_2$$

Proof Sketch: Assume by contradiction that the above does not hold. Without loss of generality, assume that there exists an i such that

$$\Pr[a_{i+1} = s] > \Pr[b_i = s] + 2\beta_1.$$

In the proof, we consider an alternative strategy σ_1^i for P_1 which is identical to the prescribed strategy σ_1 except that it instructs the party P_1 to quit before broadcasting the message in round i. Assuming that the other party (P_2) does broadcast its share in that round, and that the execution reaches round i, we have that P_1 outputs a_{i+1} while P_2 outputs b_i. Using the contradicting assumption, it follows that:

$$\Pr[a_{i+1} = s \wedge b_i \neq s] \geq \Pr[a_{i+1} = s] - \Pr[b_i = s] > 2\beta_1.$$

That is, with probability at least $2\beta_1$ the outcome is o_1^+, and therefore P_1 gains U_1^+ while P_2 gains only U_2^{--}. Thus, the expected utility of P_1 is at least

$$2\beta_1 \cdot U_1^+ + (1-2\beta_1) \cdot U_1^{--} = 2\beta_1(U_1^+ - U_1^{--}) + U_1^{--} = 2U_1 - 2U_1^{--} + U_1^{--} > U_1$$

where the last equality is by the assumption that U_i is non-negligibly greater than U^- and U_i^{--} (note that if $U_i \approx U^-$ then P_i has no reason to play at all). Thus, the strategy σ_1^i of stopping in round i is a better strategy for P_1, in contradiction to the assumption that $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ is a Nash equilibrium.

We stress that some important details are omitted from this proof sketch. For example, it does not take into account the probability that round i is actually reached in the execution or the possibility of negligible failure; see the full version for details. ■

We use the above claim to prove our lower bound. Now, consider the case that the secret is a uniformly distributed k-bit string. In such a case, the probability that any party outputs the correct secret before receiving any message is negligible (i.e., $\Pr[a_0 = s] = \Pr[b_0 = s] = \mu(k)$ for some negligible function μ). By simple induction, we have that for every i:

$$\Pr[a_i = s] \leq 2i\beta + \mu(k) \qquad \text{and} \qquad \Pr[b_i = s] \leq 2i\beta + \mu(k)$$

and so

$$\sum_{i=1}^{2r(k)} \Pr[a_i = s] \leq \sum_{i=1}^{2r(k)} 2i\beta + \mu(k) \approx 4\beta \cdot r^2(k)$$

where $r(k)$ denotes the expected number of rounds; i.e., $E[R^{\Gamma}_{(\sigma_1,\sigma_2)}] = r(k)$. By Markov, $\Pr\left[R^{\Gamma}_{(\sigma_1,\sigma_2)} \geq 2r(k)\right] \leq \frac{1}{2}$ and so $\Pr\left[R^{\Gamma}_{(\sigma_1,\sigma_2)} < 2r(k)\right] > \frac{1}{2}$. Now, if $R^{\Gamma}_{(\sigma_1,\sigma_2)} < 2r(k)$ then for some $i \in \{1, \ldots, 2r(k)\}$ it holds that $a_i = s$ (because at the end, both parties must output s). Thus,

$$\sum_{i=1}^{2r(k)} \Pr[a_i = s] > \frac{1}{2}.$$

We conclude that

$$\frac{1}{2} < \sum_{i=1}^{2r(k)} \Pr[a_i = s] \leq 4\beta r^2(k)$$

implying that $r(k) > \frac{1}{\sqrt{8\beta}}$. (Note that the theorem bounds $r(k) > \frac{1}{8\sqrt{\beta}}$ and not what we have shown here. This is due to additional factors that we have omitted from this sketch; see the full version for details.) ■

Using Theorem 10 we conclude that there do not exist U^+-independent fair reconstruction mechanisms with an expected number of rounds that is polynomial, even in the simultaneous model. In order to see this, we show that for all fixed polynomials U_i, U_i^-, U_i^{--} and $r(k)$, there exists a polynomial U_i^+ such that $r(k) < \frac{1}{8\sqrt{\beta}}$. Specifically, take $U_i^+ \geq 64r^2(k) \cdot \left(U_i - U_i^{--}\right) + U_i^{--}$. This suffices because in such a case

$$\beta_i = \frac{U_i - U_i^{--}}{U_i^+ - U_i^{--}} \leq \frac{U_i - U_i^{--}}{64r^2(k) \cdot \left(U_i - U_i^{--}\right) + U_i^{--} - U_i^{--}} = \frac{1}{64r^2(k)}$$

and thus $r(k) \leq \frac{1}{8\sqrt{\beta_i}}$ in contradiction.

4.2 Impossibility for U^f-Independence (Non-simultaneous)

In Section 3 we showed that any mechanism that is U^f-independent achieves correctness. In the simultaneous channels model, U^f-independence – and correctness – has been achieved by previous protocols [5,9]. However, as we have mentioned, the known protocols for the model with non-simultaneous channels do *not* guarantee correctness. In particular, if $U_i^f > U_i$ for some party P_i then the strategy profiles $\boldsymbol{\sigma}$ of [10,7] are *not* computational Nash equilibriums. In this section we prove that this is inherent to the non-simultaneous model. That is, there does not exist a fair reconstruction mechanism that is U^f-independent in the non-simultaneous model.

The Kol-Naor mechanism [10] and correctness. Before proceeding with our proof, we describe the mechanism of Kol and Naor for non-simultaneous channels and show why it does not achieve correctness. This example illustrates the problem of achieving U^f-independence and is thus very instructive. The Kol-Naor mechanism assumes that the utility functions $\mathcal{U}$ fulfill the assumptions in Definition 4. Furthermore, the mechanism itself is constructed given the actual values of the utility functions (i.e., it is utility dependent). The general idea of their protocol is that the shares assigned to the party are actually lists of possible secrets. One party receives a list of size ℓ (this party is called "the short party"), and the other party receives a list of size $\ell + d$ (this party is called "the long party"). The short list is a strict prefix of the other. The lengths ℓ and d are chosen according to a geometric distribution with parameter β, where β depends on the utility functions of the parties. The real secret is located at position $\ell + 1$ in the long list, while all the other elements in the lists are fake; the $(\ell + 1)$th round is called the definitive round because in this round the secret is learned. In addition to the lists described above, the dealer selects an independent random permutation for every round; this permutation determines the order in which the parties send their list elements in the round. The party that sends its message first in the definitive round is given the long list, and the other party is given the short list. In addition, the parties receive the permutations for the rounds appearing in their respective lists (i.e., the short party receives the permutation only for the first ℓ rounds). We stress that neither party knows if it the short or long party. In any given round, we call the party who sends its element first the "first party" and we call the other the "second party".

In order to reconstruct the secret, the parties proceed round by round; in the ith round each party sends its ith list element in the order determined by the permutation. At iteration $\ell + 1$ (the "definitive iteration"), the long party is the first to broadcast its share (that is, it is the "first party"). However, the short party's list is finished and thus it has no element to send. It therefore remains silent in this round. The first round in which only one party sends a list element is the definitive round, and so the secret sent in this round is taken to be the real secret. Intuitively, fairness is achieved because the owner of the long list does not know the length of the short list, and in particular does not know which round is the definitive round. It therefore does not know which of the elements in its list is the real secret and so has to send its share every round. See [10] for details.

As pointed out in [10, Note 6.2], if one of the parties aborts prematurely (i.e., remains silent in round i for some $i < \ell$) then the other party will output an incorrect value (with high probability the element s_i of the ith round will not equal the secret). It is important to note that the aborting party knows that s_i is not the real secret because its list is not yet finished. Furthermore, it can even have some influence over the incorrect value output by the first party (this is because it can choose at which point to stop and thus it can choose which of the values in the prefix of the list is output by the first party). The protocol is therefore clearly not correct. We remark that the same problem also exists for the protocol of [7]. As we have mentioned, [10] assume that rational parties will not behave in this way because they always prefer to learn the secret than to not learn it (observe that if a party aborts prematurely then it will not learn the real secret). That is, they assume that $U_i^f < U_i$. We show that this assumption is essential as long as U^f-independence is desired.

The impossibility result. Our proof of impossibility assumes that for all i, U_i^+ is strictly greater than U_i by a non-negligible amount; this is called strict competitiveness; see [10]. We are now ready to formally state the theorem.

Theorem 12. *There do not exist U^f-independent fair reconstruction mechanisms for strictly competitive utility functions in the non-simultaneous model.*

By Proposition 9, U^f-independence implies correctness. We therefore prove that in the non-simultaneous model there does not exist a fair reconstruction mechanism that is *correct*, as defined in Definition 7.

Intuition: We begin by describing 2 strategies σ_1^{stop} and σ_2^{stop}. The strategy σ_1^{stop} for party P_1 is the strategy that follows the prescribed $\boldsymbol{\sigma}$ in all the rounds with the following difference. In every round, P_1 checks what its output would be if P_2 quits at that round. In the first round for which the output is *not* $\perp$, the strategy σ_1^{stop} instructs P_1 to quit at that round. σ_2^{stop} is defined analogously. Since we assume correctness, the probability that one of the parties will output a value which is not s or $\perp$ when the other prematurely aborts is negligible. Thus, when playing σ^{stop} both of the parties will output the correct s in the round that they quit. Next, we prove that when both parties follow $\boldsymbol{\sigma}^{stop}$, with high probability one of them learns the secret while the other does not. We conclude by showing that the prescribed strategy $\boldsymbol{\sigma}$ is not a computational Nash equilibrium by showing that one of the $\boldsymbol{\sigma}^{stop}$ strategies has a better expected utility than $\boldsymbol{\sigma}$. That is, we show that either $u_2(\sigma_1, \sigma_2^{stop}) > u_2(\sigma_1, \sigma_2) + \epsilon'$ or $u_1(\sigma_1^{stop}, \sigma_2) > u_1(\sigma_1, \sigma_2) + \epsilon'$, for some non-negligible function ϵ'. The proof of this appears in the full version.

5 Positive Results

5.1 U^f-Dependent Reconstruction in the Non-simultaneous Model

In this section, we address the basic question of whether or not it is possible to construct a fair and correct reconstruction mechanism using non-simultaneous channels even if $U_i^f \geq U_i$. We answer this in the positive by constructing a mechanism that works as long as it knows the value of U_i^f for each party P_i (in the same way that the mechanism knows the values of U_i^+, U_i, U_i^- and U_i^{--}).

The idea behind the mechanism. We will consider the two party case only, but the idea works for the multiparty case as well. We assume familiarity with the protocol of Kol and Naor [10]; see the beginning of Section 4.2 for a short description of the protocol and why it does not guarantee correctness. This will be used below. Looking closely at the strategy for breaking correctness in the Kol-Naor mechanism, it arises because the first party to send its list element in an iteration has no way of verifying if the current round is the definitive round or not. This is necessary because if the long party could check if the current round is the definitive one before sending its element, it could learn the secret without the other party learning it. Despite this, our key observation is that it is not necessary that all of the fake iterations be the same, as in the Kol-Naor mechanism. Rather, we introduce additional rounds with the property that the second party in each such round knows that the round is fake while the first party does not. Now, if a first party prematurely aborts on such a round, then it will gain only U^-, and not U^f (because the second party knows that the first party has cheated and just aborts outputting $\perp$). By adding enough of these additional rounds, we have that the probability that a party successfully achieves U^f is low enough so that a higher expected utility is obtained by playing $\boldsymbol{\sigma}$ and obtaining U. See the full version for a detailed description and proof.

5.2 Full Independence for $n \geq 3$ with Relaxed Assumptions

In this section we show that utility dependence is not always essential. In particular, we show that for a certain reasonable relaxation of the utility functions, it is possible to construct a *utility independent* fair reconstruction mechanism for the case of t-out-of-n secret sharing, where $n \geq 3$. We do not claim that our assumptions always hold or should be used; rather our aim here is to show that utility independence can sometimes be achieved.

The "standard" assumptions [6,10] typically used for the utility functions are that a party always prefers to learn than not. Furthermore, assuming that a party learns, the fewer others that learn the better. We relax these assumptions, and assume that each party prefers to learn the secret alone, but once *one* of the other parties learns the secret it doesn't matter how many other parties learn it (thus U_i^+ denotes the utility when it alone learns, and U_i denotes the utility that it learns along with any positive number of other parties).

In addition to the above, we assume that the utility functions are polynomial in the security parameter, and that there is a non negligible difference between them. That is, there exists a polynomial $p(\cdot)$, such that for infinitely many k's it holds that: $U_i \geq U_i^- + \frac{1}{p(k)}$. (Our impossibility result for U^+-independence when $n = 2$ holds for such utility functions.) This is a natural extension of the strict differences between the utility functions, as defined in [10], when they are modeled as functions of the security parameter. (We remark that U_i^+ may equal U_i; we only need a non-negligible difference between U_i and U_i^-.) Note that when the above does not hold, it means that P_i's utility when not learning is essentially the same as when learning. Thus, P_i may as well not participate at all and this case is not interesting. Our protocol assumes simultaneous channels in order to achieve U^f-independence. As we showed in Theorem 12, it is impossible to achieve U^f independence with non-simultaneous channels.[2]

The protocol idea. The idea behind our protocol is to enable one of the parties to learn the secret even when the others do not. Now, once this party has learned the secret, it is not possible for any other party to obtain U^+. Thus, the other parties can either continue with the execution of the protocol and obtain U, or they can quit and obtain only U^- (which is strictly less than U by the assumption that $U_i \geq U_i^- + \frac{1}{p(k)}$). The main question is how to construct a protocol so that one of the parties can learn the secret, but only after there are $t^* \geq t$ parties participating in the reconstruction phase, but then enable the residual $t-1$ parties to reconstruct the secret without the cooperation of the party who already learned the secret.

We achieve this in the following way. Let s be the secret to be shared; for simplicity assume that $s \in \{0,1\}^k$ (where k is the security parameter). The dealer chooses a random $r \in_R \{0,1\}^k$ and generates shares of r and s with threshold t and shares of $r \oplus s$ with threshold $t-1$ (overall three sets of shares). The dealer then sends each party its shares. Before proceeding we note that no set of $t-1$ parties can reconstruct the secret s, because even though a set of this size can learn $r \oplus s$, without knowing r this is of no help. In addition, ignoring issues of rationality and utility, it is possible for every set of t parties to obtain s by just reconstructing the shares of s, or by reconstructing r and $r \oplus s$ (where the latter requires only $t-1$ to participate).

We now informally describe our reconstruction protocol. In the first phase of the protocol $t^* \geq t$ parties reconstruct r by simply sending their shares to all others. In the second phase, the t^* parties reconstruct s by sending their shares one at a time consecutively (here it is crucial that a *simultaneous* channel *not* be used and so we use the simultaneous channel as a non-simultaneous one, by having every party wait until it receives all previous messages before sending its own). Note that at the end of the second phase, the last t^*-t+1 parties can reconstruct the secret alone, and thus, they may not send their shares. If any of the parties does not send their share in the first phase, or if any of the first $t-1$ parties does not send their share in the second phase, then all parties abort and output $\perp$. At the end of this phase, unless all have aborted, there remain $t-1$ parties who have not learned the secret. These parties continue to the third phase. The crucial observation is that none of these parties can obtain

[2] A version of our protocol for the non-simultaneous model can be constructed using the techniques of [10] and our protocol in Section 5.1. However, note that the protocol for the non-simultaneous model needs to know the values of U_i^f, U_i and U_i^{--}, and therefore the result is only U^+-independent.

U^+ since there are already $t^* - t + 1 \geq 1$ parties who have learned the secret s. We utilize this fact to use any one of the known rational reconstruction protocols while setting $\beta = \frac{1}{2}$ (where β is a parameter that usually depends on the utility values, like our β in the lower bound); observe that we fix β irrespective of the actual utility values. This works because at this point, once one party has learned the secret, the maximum possible utility the parties can obtain is U. In particular, even if only one party of the remaining $t-1$ parties learns the secret, its utility is still U because one party already knows the secret. Now, the known rational secret sharing protocols with $\beta = \frac{1}{2}$ all have the property that if the parties follow $\boldsymbol{\sigma}$ then they will obtain U (with probability 1). However, if they do not, then with probability $1-\beta$ they will obtain U^-. Thus, the expected utility by not following $\boldsymbol{\sigma}$ is $\frac{1}{2} \cdot U^- + \frac{1}{2} \cdot U < U$, and so the parties follow $\boldsymbol{\sigma}$ and all learn the secret.

Remark. Our construction can be extended to deal with the case that parties *do* prefer that as few as possible other parties learn, but *do not care* whether $t-2$ or $t-1$ parties learn (i.e., it does not make any difference if all learn, or all but one learn). This is a much milder relaxation on the utility functions; see the full version for details.

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.Y.: Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation. In: The 25th PODC, pp. 53–62 (2006)
2. Cleve, R.: Limits on the Security of Coin Flips when Half the Processors are Faulty. In: 18th STOC, pp. 364–369 (1986)
3. Dodis, Y., Rabin, T.: Cryptography and Game Theory. In: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
4. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
5. Gordon, S.D., Katz, J.: Rational Secret Sharing, Revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)
6. Halpern, J., Teague, V.: Rational Secret Sharing and Multiparty Computation. In: The 36th STOC, pp. 623–632 (2004)
7. Fuchsbauer, G., Katz, J., Levieil, E., Naccache, D.: Efficient Rational Secret Sharing in the Standard Communication Model. Cryptology ePrint Archive, Report #2008/488 (2008)
8. Katz, J.: Bridging Game Theory and Cryptography: Recent Results and Future Directions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 251–272. Springer, Heidelberg (2008)
9. Kol, G., Naor, M.: Cryptography and Game Theory: Designing Protocols for Exchanging Information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg (2008)
10. Kol, G., Naor, M.: Games for Exchanging Information. In: The 40th STOC, pp. 423–432 (2008)

11. Izmalkov, S., Micali, S., Lepinski, M.: Rational Secure Computation and Ideal Mechanism Design. In: The 46th FOCS, pp. 585–595 (2005)
12. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely Fair SFE and Coalition-Safe Cheap Talk. In: The 23rd PODC, pp. 1–10 (2004)
13. Lepinski, M., Micali, S., Shelat, A.: Collusion-Free Protocols. In: The 37th STOC, pp. 543–552 (2005)
14. Lysyanskaya, A., Triandopoulos, N.: Rationality and Adversarial Behavior in Multiparty Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 180–197. Springer, Heidelberg (2006)
15. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
16. Ong, S.J., Parkes, D., Rosen, A., Vadhan, S.: Fairness with an Honest Minority and a Rational Majority. In: The 6th TCC. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)

On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem

Vadim Lyubashevsky[1,⋆] and Daniele Micciancio[2,⋆⋆]

[1] School of Computer Science, Tel Aviv University
Tel Aviv 69978, Israel
vlyubash@cs.ucsd.edu

[2] Computer Science and Engineering Department,
University of California at San Diego,
La Jolla, CA 92093, USA
daniele@cs.ucsd.edu

Abstract. We prove the equivalence, up to a small polynomial approximation factor $\sqrt{n/\log n}$, of the lattice problems uSVP (unique Shortest Vector Problem), BDD (Bounded Distance Decoding) and GapSVP (the decision version of the Shortest Vector Problem). This resolves a long-standing open problem about the relationship between uSVP and the more standard GapSVP, as well the BDD problem commonly used in coding theory. The main cryptographic application of our work is the proof that the Ajtai-Dwork ([2]) and the Regev ([33]) cryptosystems, which were previously only known to be based on the hardness of uSVP, can be equivalently based on the hardness of worst-case $\mathrm{GapSVP}_{O(n^{2.5})}$ and $\mathrm{GapSVP}_{O(n^2)}$, respectively. Also, in the case of uSVP and BDD, our connection is very tight, establishing the equivalence (within a small constant approximation factor) between the two most central problems used in lattice based public key cryptography and coding theory.

1 Introduction

Lattice based cryptography is among the most compelling alternatives to traditional methods based on number theory. Ajtai's ground-breaking discovery that lattice problems exhibit a worst-case to average-case connection [1] immediately yielded one-way functions and collision resistant hash functions based on the worst-case hardness of several lattice approximation problems, and prompted researchers to investigate the construction of more complex cryptographic primitives (most notably public key encryption) based on lattices. The first cryptosystem that was based on the worst-case hardness of lattice problems was

⋆ Supported by the Israel Science Foundation and by a European Research Council (ERC) Starting Grant.

⋆⋆ Supported in part by NSF grant CCF-0634909. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

the Ajtai-Dwork cryptosystem [2]. The security of this system was based on the worst-case hardness of the approximate "unique" Shortest Vector Problem uSVP$_{O(n^8)}$ (in uSVP$_\gamma$, we are asked to find the shortest vector in a lattice in which the shortest vector is guaranteed to be at least γ times smaller than the next shortest non-parallel lattice vector). This was followed by an improvement to their cryptosystem [11], and the currently best version of it is based on the hardness of uSVP$_{O(n^2)}$. In a later work, Regev built a different cryptosystem based on worst-case uSVP$_{O(n^{1.5})}$ [33]. But while other cryptographic primitives could be built on the hardness of the more general, and better understood from a complexity-theoretic point of view, shortest vector problem on general lattices (in its decision variant, GapSVP), cryptosystems seemed to require the hardness of the potentially easier uSVP lattices. So it was a major open problem as to whether lattice-based cryptosystems could be based on the hardness of problems on general lattices, and GapSVP in particular. What made the problem even more interesting was that simpler cryptographic primitives from "minicrypt"[1] such as one-way functions [1], collision-resistant hash functions [1,10,29] identification schemes [18,25,30] and signature schemes [8,26] could be based on the worst-case hardness of GapSVP.

A breakthrough in the design of lattice-based cryptosystems, in the sense of deviating from uSVP, came when Regev built a cryptosystem which was actually based on GapSVP (as well as some other standard lattice problems), but the assumption was that approximating GapSVP was hard even by quantum algorithms [35]. Another breakthrough came just recently, when Peikert finally constructed a cryptosystem that is based on the hardness of GapSVP under classical reductions [32]. Of course, a different way of obtaining cryptosystems with security based on GapSVP would be to establish a relation between GapSVP and uSVP, and this is precisely what we do in this paper.

On the practical front, about at the same time as Ajtai's discovery [1], two cryptosystems were proposed (GGH [12] and NTRU [15]), which, while lacking a security proof from worst-case lattice assumptions, are intuitively very appealing. These cryptosystems rest on the conjectured average-case hardness of the bounded distance decoding problem (BDD), which can be considered a special version of the closest vector problem, very much like uSVP is a special version of the shortest vector problem. Additionally, Regev's cryptosystem [35] whose security is based on the worst-case hardness of quantum GapSVP is equivalently based on an average-case version of classical BDD (used in [35] under the name "Learning with Errors" problem.) So, the average-case BDD problem seems quite a natural problem to consider in the setting of lattice based public key encryption.

Our contribution. In this paper, we prove the equivalence, up to a factor of $\sqrt{n/\log n}$, of the GapSVP, BDD, and uSVP problems. In particular, we prove

[1] Minicrypt [16] consists of all cryptographic primitives that can be derived from one-way functions, or more generally, exist relative to a random oracle. Collision resistant hash functions are not known to be reducible to one-way functions, but still exist relative to a random oracle, so they can be included in minicrypt.

Cryptosystem	GapSVP Approximation Factor	Message Expansion
Ajtai-Dwork [2]	$\mathbf{\tilde{O}(n^{2.5})}$	$O(n^2)$
Regev [33]	$\mathbf{\tilde{O}(n^2)}$	$O(n)$
Peikert [32]	$\tilde{O}(n^2)$	$O(\log n)$

Fig. 1. Cryptosystems based on worst-case GapSVP$_\gamma$. The results in bold-face are consequences of the current work.

that for any $\gamma \geq 1$, there is a reduction from BDD$_{1/2\gamma}$ to uSVP$_\gamma$, and for any polynomially-bounded γ, there is a reduction from uSVP$_\gamma$ to BDD$_{1/\gamma}$. (We remark that the BDD$_\alpha$ problem is easier for *smaller* values of the factor α, while uSVP$_\gamma$ is easier for *larger* values of γ. For a formal definition of the problems, see the next section.) So, the problems uSVP$_\gamma$ and BDD$_{1/\gamma}$ are essentially equivalent under polynomial time reduction that preserve the approximation factor up to a small constant $\gamma/\gamma' \leq 2$. We also show reductions from uSVP$_\gamma$ to GapSVP$_\gamma$, and from GapSVP$_\gamma$ to BDD$_{\frac{1}{\gamma}\sqrt{n/\log n}}$ (for any $\gamma > 2\sqrt{n/\log n}$).

So, in summary, all three problems uSVP$_\gamma$, BDD$_{1/\gamma}$ and GapSVP$_\gamma$ are equivalent up to polynomial approximation factors, and all currently known lattice based public key cryptosystems with classical worst-case security guarantees [2,32,33] are qualitatively equivalent. In particular, our results imply that the Ajtai-Dwork [2] and the Regev [33] cryptosystems are based on the hardness of GapSVP$_{\tilde{O}(n^{2.5})}$ and GapSVP$_{\tilde{O}(n^2)}$ respectively. And since Peikert's recent cryptosystem [32] is also based on the hardness of the GapSVP$_{\tilde{O}(n^2)}$, the only major quantitative difference between the three cryptosystems is that Peikert's has a smaller message expansion factor (see Figure 1 and also [32] for more details).

When it comes to the practical GGH [12] and NTRU [15] cryptosystems, we cannot formally draw any implications from our findings, because ours are worst-case to worst-case reductions, and the GGH and NTRU cryptosystems lack security proofs from worst-case problems. Still, our results show that the (average-case) BDD lattice problems underlying GGH and NTRU, and those used in more theoretical constructions, have much more in common than previously thought.

In addition to cryptographic applications, the uSVP problem also found applicability in areas of learning theory and quantum computation. Klivans and Sherstov showed that a polynomial-time algorithm PAC-learning the intersection of n^ϵ half-spaces implies a polynomial-time algorithm for solving uSVP [20]. Regev showed that a solution to the dihedral coset problem would imply a quantum algorithm for uSVP [34]. Our work implies that the two problems above are based on the more well-studied GapSVP problem. This seems especially important for Regev's result since there was very little prior evidence that uSVP was hard for quantum computers.

1.1 Previous Work

There has been a lot of work in establishing relationships between various lattice problems. In fact, while our central cryptographic result (that the Ajtai-Dwork and the Regev cryptosystems are based on the hardness of GapSVP) is new, the components that comprise it are very much based on prior work.

Proving the reduction from GapSVP to uSVP can be broken down into two separate reductions. In section 4, we give a reduction from BDD to uSVP and in section 7 we give a reduction from GapSVP to BDD. The BDD to uSVP reduction uses an idea that dates back to at least the classic result of Lagarias and Olyzsko [22] where random low-density subset sum instances are converted to lattices with a unique shortest vector. This same idea has subsequently been used in various guises in reductions [6,17] as well as in heuristic attacks on cryptographic primitives [31].

The reduction from GapSVP to BDD is already implicit in the recent work of Peikert [32]. And in fact, almost the same idea was already used in the work of Goldreich and Goldwasser [9] where it was proved that GapSVP (and other lattice problems) are in the complexity class coAM. In that work, an all-powerful prover was able to convince a polynomially-bounded verifier that the length of the shortest vector of the lattice is large. The GapSVP to BDD reduction is obtained by realizing that the all-powerful prover in the coAM protocol can simply be substituted with a BDD oracle.

The other two reductions presented in our work are also related to some previous works. The reduction from uSVP to BDD in section 5 uses some ideas from the SVP to CVP reduction of Goldreich, et al. [13]. The reduction from uSVP to GapSVP is based on Regev's reduction from the decision to the search version of uSVP [33], but our proof is somewhat simpler and tighter.

1.2 Discussion and Open Problems

As mentioned earlier, one of the separations between the lattice-based "minicrypt" primitives and lattice-based public key cryptosystems was that the former could be based on the hardness of classical GapSVP, whereas the latter could not. But our work, as well as the recent work of Peikert [32], shows that there are cryptosystems based on the worst-case hardness of the shortest vector problem in its decision version. Nevertheless, there still seems to be a difference in the types of problems that "minicrypt" primitives can be based on and the hardness assumptions needed for public-key cryptosystems. The aforementioned "minicrypt" primitives [1,8,10,18,25,26,29,30] can all be based on a standard lattice *search* problem SIVP, in addition to GapSVP. We remark that, up to a polynomial loss in the approximation factor, GapSVP, uSVP and BDD can be reduced to SIVP.[2] Moreover the "quantum step" of [35] gives a *quantum* reduction from $\mathrm{SIVP}_{O(n\gamma)}$ to $\mathrm{BDD}_{1/\gamma}$. So, under quantum reductions, all lattice problems uSVP,

[2] This can be done in a variety of ways. For example, one can first reduce $\mathrm{GapSVP}_{n\gamma}$ to $\mathrm{GapSIVP}_{\gamma}$ using transference theorems, and then use a trivial reduction from $\mathrm{GapSIVP}_{\gamma}$ to SIVP_{γ}.

BDD, GapSVP, SIVP are qualitatively equivalent, up to polynomial approximation factors. However, there is no known classic polynomial time reduction from SIVP to any of uSVP, BDD, GapSVP (except in trivial cases). We also remark that the two most famous lattice problems, SVP and CVP, are equivalent under polynomial time reduction up to polynomial approximation factors [17], and there is an approximation preserving reduction from SIVP to CVP [27]. However, there is no known reduction in the opposite direction, from SVP or CVP to SIVP. So once again, this raises the question of whether lattice-based public key cryptosystems require qualitatively stronger assumptions than simpler cryptographic primitives (e.g., quantum hardness of SIVP, rather than just classic hardness), and whether cryptography in general can be based on the worst-case hardness of SVP or CVP in their search version.

It is interesting to point out that even though the cryptosystems described in [2,32,33] are all based on the hardness of GapSVP, the construction of Peikert's cryptosystem is quite different from the other two. The Regev and Ajtai-Dwork cryptosystems were based directly on the uSVP problem, while Peikert's cryptosystem is actually quite similar to the other Regev cryptosystem [35] whose hardness is based on BDD. At this point, cryptosystems based on BDD are more efficient since their message expansion factor is smaller (see Figure 1), but perhaps the connection between GapSVP, BDD, and uSVP demonstrated in this work can be somehow exploited in order to combine the two seemingly distinct techniques for cryptosystem construction and build one that is even more efficient and still based on the hardness of GapSVP.

Another outstanding question on the complexity of lattice problems is whether the search and length estimation/decision versions of the shortest vector problem are computationally equivalent. A search to decision reduction for the approximate SVP would immediately imply the equivalence (up to polynomial factors) of all lattice problems uSVP, BDD, GapSVP, SVP, CVP, SIVP considered in cryptography.

There are also many questions about the relationship between lattice problems that are raised directly from our work. One such problem is whether the reductions in sections 5 and 6 can be extended to approximation factors that are not restricted to being polynomial. Another problem is to figure out whether the small gap that we have in the connection between BDD and uSVP can be closed. At this point, we have the reduction $\text{uSVP}_\gamma \leq \text{BDD}_{1/\gamma} \leq \text{uSVP}_{\gamma/2}$, which is loose by a factor of 2. We believe that there are three (mutually exclusive) possibilities for possible improvements of this result. It might be possible to show that:

1. $\text{uSVP}_{\gamma/2} \leq \text{BDD}_{1/\gamma}$ or
2. $\text{BDD}_{1/\gamma} \leq \text{uSVP}_\gamma$ or
3. $\text{uSVP}_\gamma \leq \text{BDD}_{\sqrt{2}/\gamma} \leq \text{uSVP}_\gamma$

This open problem also has an intriguing connection with the computational complexity of uSVP_γ. Unlike SVP_γ, which is known to be NP-hard for any constant γ, uSVP_γ is only NP-hard for $\gamma = 1 + 2^{-n^c}$ for some constant c [21].

So proving the NP-hardness of uSVP$_\gamma$ for larger factors is a very interesting open problem. One possibility for doing so would be to prove item (2) above. Combining this with the result of [24] that states that BDD$_\gamma$ is NP-hard for $\gamma > 1/\sqrt{2}$, we would obtain that uSVP$_\gamma$ is NP-hard for $\gamma = \sqrt{2}$.

The possibility of somehow using the reduction from GapSVP to uSVP in order to prove NP-hardness of uSVP is also intriguing. At this point, the reduction requires the γ in GapSVP$_\gamma$ to be at least $\sqrt{n/\log n}$, and the GapSVP$_\gamma$ problem is not NP-hard for such parameters unless the polynomial-time hierarchy collapses. While there seem to be some technical roadblocks for reducing this requirement, it is not entirely clear that this should not be possible.

1.3 Organization of the Paper

Those readers interested mainly in the reduction from GapSVP to uSVP (which implies that the security of the Ajtai-Dwork and Regev cryptosystems is based on worst-case GapSVP) can simply read sections 4 and 7 for the proofs of BDD $\leq$ uSVP and GapSVP $\leq$ BDD respectively. Section 4 uses a result from section 3 in order to strengthen the reduction a bit, but this can be safely skipped.

In order to establish the equality of the three problems, we also need to prove that uSVP $\leq$ BDD and uSVP $\leq$ GapSVP. This is done in sections 5 and 6 respectively.

2 Preliminaries

An n-dimensional *lattice* is a discrete additive subgroup of $\mathbb{R}^n$. A set of linearly independent vectors that generates a lattice is called a basis, and we will denote it as an $n \times m$ matrix $\mathbf{B}$ whose m columns $\mathbf{b}_i$ are the generating vectors. The lattice generated by the basis $\mathbf{B}$ will be written as $\mathcal{L}(\mathbf{B})$. The *span* of a basis $\mathbf{B}$, denoted $Span(\mathbf{B})$, is the collection of all points $\mathbf{By}$ where $\mathbf{y} \in \mathbb{R}^m$. The *fundamental parallelepiped* of an $n \times m$ basis $\mathbf{B}$, written as $\mathcal{P}(\mathbf{B})$, is defined as the collection of all points that can be written as $\mathbf{By}$ where $\mathbf{y} \in [0,1)^m$. Every point $\mathbf{s} \in \mathbb{R}^n$ has a unique associated point $\mathbf{t}$ inside $\mathcal{P}(\mathbf{B})$ such that $\mathbf{s} = \mathbf{t}$ in the quotient group $Span(\mathbf{B})/\mathcal{L}(\mathbf{B})$. This point is denoted $\mathbf{t} = \mathbf{s} \bmod \mathbf{B}$ and can be computed from $\mathbf{s}$ in polynomial time. For any point $\mathbf{t}$, in $\mathbb{R}^n$ and any lattice $\mathcal{L}(\mathbf{B})$, the distance of $\mathbf{t}$ to the lattice is written as $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$.

For any point $\mathbf{t} \in \mathbb{R}^n$ and $r \in \mathbb{R}$, let $\mathcal{B}(\mathbf{t}, r)$ denote a ball of radius r centered at $\mathbf{t}$. The *shortest vector* of a lattice $\mathcal{L}(\mathbf{B})$ is the non-zero vector in $\mathcal{L}(\mathbf{B})$ with the smallest ℓ_2 norm. The length of the shortest vector, referred to as the *minimum distance*, of $\mathcal{L}(\mathbf{B})$ is denoted by $\lambda_1(\mathcal{L}(\mathbf{B}))$ (or $\lambda_1(\mathbf{B})$ for short). The notion of minimum distance can be generalized to define the i^{th} successive minimum $\lambda_i(\mathbf{B})$ as the smallest radius r such that $\mathcal{B}(\mathbf{0}, r)$ contains i linearly independent lattice points. The determinant of a lattice $\mathcal{L}(\mathbf{B})$ is defined as $\sqrt{det(\mathbf{B}^T\mathbf{B})}$. When $\mathbf{B}$ is a full-rank lattice, the previous definition becomes just $|det(\mathbf{B})|$. Lattices $\mathcal{L}(\mathbf{B})$ and $\mathcal{L}(\mathbf{D})$ are called *dual* if $\mathcal{L}(\mathbf{D}) = \{\mathbf{y} : \forall \mathbf{v} \in \mathcal{L}(\mathbf{B}), \mathbf{y} \cdot \mathbf{v} \in \mathbb{Z}\}$. If $\mathcal{L}(\mathbf{B})$ and

$\mathcal{L}(\mathbf{D})$ are duals, then $det(\mathcal{L}(\mathbf{B})) = det(\mathcal{L}(\mathbf{D}))^{-1}$. Minkowski's theorem states that for any n-dimensional lattice $\mathcal{L}(\mathbf{B})$, $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \sqrt{n} \cdot det(\mathcal{L}(\mathbf{B}))^{1/n}$. For additional information about lattices, please refer to [28].

2.1 GapSVP

Possibly the most well-known lattice problem is the Shortest Vector Problem (SVP). It comes in both decisional and search versions, but in this paper we are only interested in the decision version. (The decision version of the problem is sometimes referred to as the *Minimum Distance Problem*). The approximation version of decisional SVP can be defined as a "gap" problem GapSVP$_\gamma$. In the GapSVP$_\gamma$ problem, we are given a basis $\mathbf{B}$ and a real number d, and are required to return YES if $\lambda_1(\mathcal{L}(\mathbf{B})) \leq d$, and return NO if $\lambda_1(\mathcal{L}(\mathbf{B})) > \gamma d$. If $\lambda_1(\mathcal{L}(\mathbf{B}))$ falls between d and γd, we can return anything. The GapSVP$_\gamma$ problem is NP-hard for any constant γ [19,14]. The fastest algorithm for solving GapSVP$_\gamma$ for $1 \leq \gamma \leq poly(n)$ takes time $2^{O(n)}$ [3]. Using the LLL algorithm [23], it is possible to find a vector that has length at most $2^{n/2}\lambda_1(\mathbf{B})$ in polynomial time.

2.2 uSVP and BDD

We now give precise definitions for the other two lattice problems that are central to this work. We urge the reader to notice that while the minimum distance problem described in the previous section is a *decision* problem, the ones in this section are *search* problems.

Definition 1 (γ-unique Shortest Vector (uSVP$_\gamma$)). *Given a lattice* $\mathbf{B}$ *such that* $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$, *find a nonzero vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *of length* $\lambda_1(\mathbf{B})$.

Definition 2 (α-Bounded Distance Decoding (BDD$_\alpha$)). *Given a lattice basis* $\mathbf{B}$ *and a vector* $\mathbf{t}$ *such that* $\operatorname{dist}(\mathbf{t}, \mathbf{B}) < \alpha\lambda_1(\mathbf{B})$, *find the lattice vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *closest to* $\mathbf{t}$.

The uSVP$_\gamma$ problem is known to be NP-hard when $\gamma = 1 + 2^{-n^c}$ for some constant c [21], and it's an outstanding open problem whether NP-hardness can be proved for larger γ. There has been some evidence to suggest that uSVP is easier than the search version of SVP [7], and that approximating the length of the shortest vector in lattices with a unique shortest vector may be easier than GapSVP in general lattices [5].

The BDD$_\alpha$ problem has been shown to be NP-hard for $\alpha > 1/\sqrt{2}$ [24] and it is an open problem whether it's hard for smaller α. We would just like to draw the reader's attention to the fact that the BDD$_\alpha$ problem becomes *harder* as α becomes larger, while the uSVP$_\gamma$ problem becomes *easier* as γ increases. Sometimes uSVP$_\gamma$ and BDD$_\alpha$ are defined in a more relaxed way, as follows:

Definition 3 (uSVP$'_\gamma$). *Given a lattice* $\mathbf{B}$ *such that* $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$, *find a nonzero vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *of length* $\|\mathbf{v}\| \leq \gamma\lambda_1(\mathbf{B})$.

Definition 4 (BDD'_α)**.** *Given a lattice basis* $\mathbf{B}$ *and a vector* $\mathbf{t}$ *such that* $\mathrm{dist}(\mathbf{t}, \mathbf{B}) < \alpha\lambda_1(\mathbf{B})$*, find a lattice vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{v} - \mathbf{t}\| < \alpha\lambda_1(\mathbf{B})$*.*

However, these relaxed variants are not any easier to solve than uSVP_γ and BDD_α as defined in this paper.

Lemma 1. *For any* $\gamma \geq 1$*, the problems* uSVP_γ *and* uSVP'_γ *are equivalent under polynomial time reductions.*

Proof. Clearly, uSVP'_γ reduces to uSVP_γ because any solution to uSVP_γ instance $\mathbf{B}$ is also a relaxed solution to $\mathbf{B}$ as a uSVP'_γ instance. In the other direction, let $\mathbf{B}$ be a uSVP_γ instance. Using a uSVP'_γ oracle we can find a nonzero lattice vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ of length $\|\mathbf{v}\| \leq \gamma\lambda_1(\mathbf{B})$. Using the uSVP restriction $\gamma\lambda_1 < \lambda_2$, we get that the shortest nonzero vector in $\mathcal{L}(\mathbf{B})$ must be of the form $c\mathbf{v}$ (for $c \in \mathbb{R}$), and can be easily found solving a 1-dimensional SVP instance $\mathcal{L}(\mathbf{B}) \cap \mathbf{v}\mathbb{R}$. □

The equivalence between BDD_α and BDD'_α is a bit trickier.

Lemma 2. *For any* $\alpha \geq 1$*, the problems* BDD_α *and* BDD'_α *are equivalent under polynomial time reductions.*

Proof. As in the previous theorem, the reduction from BDD'_α to BDD_α is trivial. Reducing BDD_α to BDD'_α is also trivial when $\alpha \leq 1/2$, because there is at most one lattice point within distance $\lambda_1(\mathbf{B})/2$ from any target. When $\alpha > 1/2$ the reduction is not as trivial because the BDD'_α oracle may return one of several lattice points, at distance from the target ranging from $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$ to $\alpha\lambda_1(\mathbf{B})$. This technical problem can be easily solved as follows. Let $(\mathbf{B}, \mathbf{t})$ be a BDD_α instance, and assume without loss of generality that $\mathbf{B}$ and $\mathbf{t}$ have integer entries. Consider the BDD'_α instance $(\mathbf{B}', \mathbf{t}')$ where

$$\mathbf{B}' = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0}^T & d/\alpha \end{bmatrix} \qquad \mathbf{t}' = \begin{bmatrix} \mathbf{t} \\ 0 \end{bmatrix}$$

for some $d > 0$. Notice that we still have $\mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}')) = \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$. In fact, the extra coordinate and basis vector in $\mathbf{B}'$ have the only effect of reducing the length of the shortest vector in the lattice to $\lambda_1(\mathbf{B}') = \min(\lambda_1(\mathbf{B}), d/\alpha)$. Let $\mu = \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$. Then using lattice reduction algorithms (see Lemma 3) we can efficiently compute a lattice point $\mathbf{v}$ at distance $d_0 = \|\mathbf{v} - \mathbf{t}\| \in [\mu, 2^n\mu]$ from $\mathbf{t}$. If $d_0 = \mu$, then we have found the closest lattice point. So, assume $d_0 > \mu$. Notice that when $d = d_0$, the instance $(\mathbf{B}', t')$ satisfies

$$\alpha\lambda(\mathbf{B}') = \min\{d, \alpha\lambda(\mathbf{B})\} > \mu.$$

So, on input $(\mathbf{B}', \mathbf{t}')$, the BDD'_α oracle returns a lattice point $\mathbf{B}'\mathbf{z}'$ such that $\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\| < \alpha\lambda(\mathbf{B}') \leq d$. On the other hand, if $d = d_0/2^n$, then for any lattice point $\mathbf{B}'\mathbf{z}'$, we have $\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\| \geq \mu \geq d$. Using binary search, we can find a d_1 such that the BDD'_α oracle returns a lattice vector $\mathbf{B}'\mathbf{z}'$ such that $\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\| < d$

when $d = d_1\sqrt{1 + 1/d_0^2}$, but not when $d = d_1$. Without loss of generality, we can assume $\mathbf{z}' = [\mathbf{z}^T, 0]$ and $\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\| = \|\mathbf{B}\mathbf{z} - \mathbf{t}\|$.

We claim that $d_1 \leq \mu$, and therefore, since $\|\mathbf{B}\mathbf{z} - \mathbf{t}\|^2$ and μ^2 are integers,

$$\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\|^2 = \|\mathbf{B}\mathbf{z} - \mathbf{t}\| \leq \lfloor d_1^2(1 + d_0^{-2})\rfloor \leq \lfloor \mu^2 + (\mu/d_0)^2\rfloor = \mu^2.$$

So, $\mathbf{B}\mathbf{z}$ is the lattice vector closest to $\mathbf{t}$.

In order to prove the claim, assume for contradiction that $d_1 > \mu$. Then, when $d = d_1$, $\alpha\lambda(\mathbf{B}') = \min(d_1, \alpha\lambda(\mathbf{B})) > \mu$. So, the BDD'_α promise is satisfied, and on input $(\mathbf{B}', \mathbf{t}')$, the BDD'_α oracle returns a lattice point $\mathbf{B}'\mathbf{z}'$ such that $\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\| < \alpha\lambda(\mathbf{B}') \leq d_1$. This is a contradiction because we had assumed the oracle returned a lattice point such that $\|\mathbf{B}'\mathbf{z}' - \mathbf{t}'\| \geq d_1$. □

2.3 Useful Lemmas

The first lemma, due to Babai [4], states that for any point in space, we can approximate the lattice point closest to it within a factor of 2^n.

Lemma 3. *There exists a polynomial-time algorithm that, given* $\mathbf{t} \in \mathbb{R}^n$ *and a lattice* $\mathcal{L}(\mathbf{B})$*, outputs a lattice vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{v} - \mathbf{t}\| \in [\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})), \leq 2^n \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))]$.

The second lemma, due to Goldreich and Goldwasser [9], states that two spheres of large radii whose centers are relatively close to each other will have a relatively large (non-negligible) intersection.

Lemma 4. *Let* $\mathbf{x}$ *be a vector in* $\mathbb{R}^n$ *such that* $\|\mathbf{x}\| \leq d$*. If* $\mathbf{s}$ *is a point chosen uniformly at random from* $\mathcal{B}(0, d\sqrt{n/\log n})$*, then with probability* $\delta > 1/n^c$ *for some constant* c*,* $\|\mathbf{s} - \mathbf{x}\| \leq d\sqrt{n/\log n}$*.*

3 BDD Self-reduction

In this section, we will show that there is a polynomial-time Cook reduction from solving BDD_α to the slightly easier problem $\mathrm{BDD}_{\alpha(1-1/n)^c}$ for any constant c. This reduction will be used to eliminate losses of small factors in our reductions that involve BDD.

Lemma 5. *For any* $\alpha \geq 1$*, there is a polynomial-time Cook reduction from* BDD'_α *to* $\mathrm{BDD}'_{\alpha\sqrt{1-1/2n}}$.

Proof. Let $(\mathbf{B}, \mathbf{t})$ be an instance of BDD'_α, and $\mathbf{y}$ be a vector in $\mathcal{L}(\mathbf{B})$ such that $\|\mathbf{t} - \mathbf{y}\| < \alpha\lambda_1(\mathbf{B})$. We do not know the actual distance $D = \|\mathbf{t} - \mathbf{y}\|$, but we can guess an approximation d such that

$$\frac{D}{2+\sqrt{2}} \leq d \leq \frac{D}{2-\sqrt{2}} \tag{1}$$

in polynomially many tries (this follows from Lemma 3 since we can approximate D to within a factor of 2^n). Consider the set

$$S = \left\{ \mathbf{t} - \frac{jd}{\sqrt{n}}\mathbf{u}_i : i \in \{1,\ldots,n\}, j \in \{-1,1\} \right\}$$

where $\mathbf{u}_i$ is a vector with a 1 in the i^{th} position, and 0's everywhere else. We will show that this set contains a vector $\mathbf{t}'$ such that $\|\mathbf{t}' - \mathbf{y}\| \le \|\mathbf{t} - \mathbf{y}\|\sqrt{1 - 1/2n}$, which would imply that $(\mathbf{B}, \mathbf{t}')$ is an instance of $\mathrm{BDD}'_{\alpha\sqrt{1-1/2n}}$. And therefore solving polynomially many (we need to find a d in the correct range as well as try all $2n$ possibilities in the set S) instances of $\mathrm{BDD}'_{\alpha\sqrt{1-1/2n}}$ would result in a solution to BDD'_α.

Without loss of generality we can assume that $\mathbf{y} = \mathbf{0}$. Then $\|\mathbf{t}\|^2 = \sum_i t_i^2 = D^2$, and there must exist an i such that $|t_i| \ge \frac{D}{\sqrt{n}}$. Then for a $j \in \{-1,1\}$ that has the same sign as t_i, the vector $\mathbf{t}' = \mathbf{t} - \frac{jd}{\sqrt{n}}\mathbf{u}_i$ is in S and

$$\begin{aligned}\|\mathbf{t}'\|^2 &= \|(t_1,\ldots,t_{i-1},t_i - \frac{jd}{\sqrt{n}}, t_{i+1},\ldots,t_n)\|^2\\ &= D^2 - \frac{2|t_i|d}{\sqrt{n}} + \frac{d^2}{n} \le D^2 - \frac{2dD}{n} + \frac{d^2}{n} \le D^2\left(1 - \frac{1}{2n}\right)\end{aligned}$$

where the last inequality follows from (1). □

Notice that the above lemma cannot be combined with itself to obtain a reduction from BDD'_α to BDD'_β for an arbitrarily small β. This is because if $\beta = \alpha\left(\sqrt{1-1/2n}\right)^c$, we would need to solve $poly(n)^c$ instances of BDD'_β in order to solve one instance of BDD'_α. This is doable in polynomial time only if c is a constant, which leads to the following corollary:

Corollary 1. *For any $\alpha \ge 1$ and any constant c, there is a polynomial-time Cook reduction from BDD'_α to $\mathrm{BDD}'_{\alpha(1-1/n)^c}$.*

Combining the above corollary with Lemma 2, we obtain:

Corollary 2. *For any $\alpha \ge 1$ and any constant c, there is a polynomial-time Cook reduction from BDD_α to $\mathrm{BDD}_{\alpha(1-1/n)^c}$.*

4 Reducing BDD to uSVP

In this section we present the reduction from from the BDD problem to uSVP. Given an instance $(\mathbf{B}, \mathbf{t})$ of BDD, we construct a uSVP instance as in (2), where μ is the approximate distance from $\mathbf{t}$ to $\mathcal{L}(\mathbf{B})$ (we do not know this distance, but can guess a good-enough approximation). The idea is that if $(\mathbf{B}, \mathbf{t})$ is an instance of $\mathrm{BDD}_{1/(2\gamma)}$ for $\gamma \ge 1$, then the lattice $\mathcal{L}(\mathbf{B}')$ has a γ-unique shortest vector and this vector is formed by using the last column of $\mathbf{B}'$ exactly once. Therefore finding this shortest vector allows us to find the closest vector in $\mathcal{L}(\mathbf{B})$ to $\mathbf{t}$.

Theorem 1. *For any $\gamma \geq 1$, there is a polynomial time Cook-reduction from* $\mathrm{BDD}_{1/(2\gamma)}$ *to* uSVP_γ.

Proof. Let $(\mathbf{B}, \mathbf{t})$ be an instance of $\mathrm{BDD}_{1/(2\gamma)}$ and let $\mu = \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) < \lambda_1(\mathbf{B})/(2\gamma)$. Let $\mathbf{v}$ be a vector in $\mathcal{L}(\mathbf{B})$ such that $\|\mathbf{t} - \mathbf{v}\| = \mu$. The goal of the reduction is to use a uSVP_γ oracle to find $\mathbf{v}$. For simplicity, we will assume that μ is known (we will explain how to deal with this issue at the end of the proof), and define the matrix

$$\mathbf{B}' = \begin{bmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0}^T & \mu \end{bmatrix} \tag{2}$$

We will show that the lattice $\mathcal{L}(\mathbf{B}')$ contains a γ-unique shortest vector $\mathbf{v}' = [(\mathbf{v} - \mathbf{t})^T, -\mu]^T$, and therefore finding such a vector will recover the vector $\mathbf{v}$, which is the solution to the BDD instance. The length of $\mathbf{v}'$ is $\sqrt{\mu^2 + \mu^2} = \sqrt{2}\mu$ and so we need to show that all other vectors in $\mathcal{L}(\mathbf{B}')$ that are not multiples of $\mathbf{v}'$ have length at least $\lambda_1(\mathbf{B})/\sqrt{2} > \sqrt{2}\gamma\mu$.

Assume for the sake of contradiction that $\mathbf{w}'$ is a vector in $\mathcal{L}(\mathbf{B}')$ of length less than $\lambda_1(\mathbf{B})/\sqrt{2}$ that is not a multiple of $\mathbf{v}'$. We can rewrite the vector $\mathbf{w}' = [(\mathbf{w} - \beta\mathbf{t})^T, -\beta\mu]^T$ where $\beta \geq 0$ and $\mathbf{w} \in \mathcal{L}(\mathbf{B})$, and so

$$\frac{\lambda_1(\mathbf{B})}{\sqrt{2}} > \|\mathbf{w}'\| = \sqrt{\|\mathbf{w} - \beta\mathbf{t}\|^2 + (\beta\mu)^2},$$

which implies that $\beta\mu < \lambda_1(\mathbf{B})/\sqrt{2}$ and

$$\|\mathbf{w} - \beta\mathbf{t}\| < \sqrt{\frac{\lambda_1(\mathbf{B})^2}{2} - (\beta\mu)^2}.$$

Now consider the vector $\mathbf{w} - \beta\mathbf{v} \in \mathcal{L}(\mathbf{B})$. Since we assumed that $\mathbf{w}'$ was not a multiple of $\mathbf{v}'$, the vector $\mathbf{w} - \beta\mathbf{v}$ is a non-zero lattice vector. To get the contradiction, we will show that the length of this vector is strictly less than $\lambda_1(\mathbf{B})$. Using the triangular inequality, we rewrite

$$\|\mathbf{w} - \beta\mathbf{v}\| = \|\mathbf{w} - \beta\mathbf{t} - \beta(\mathbf{v} - \mathbf{t})\| \leq \|\mathbf{w} - \beta\mathbf{t}\| + \beta\|\mathbf{v} - \mathbf{t}\| < \sqrt{\frac{\lambda_1(\mathbf{B})^2}{2} - (\beta\mu)^2} + \beta\mu.$$

The last term of the above inequality is maximized when $\beta = \lambda_1(\mathbf{B})/(2\mu)$, and therefore for all β,

$$\|\mathbf{w} - \beta\mathbf{v}\| < \sqrt{\frac{\lambda_1(\mathbf{B})^2}{2} - (\beta\mu)^2} + \beta\mu \leq \lambda_1(\mathbf{B}),$$

which gives us the contradiction.

We now discuss the issue of guessing the μ such that $\mu = \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$. While we cannot guess such a μ exactly, we can, in polynomial time, guess a μ such that $(1 - 1/n)\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq \mu \leq (1 + 1/n)\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$. We can do this because we can find a d such that $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq d \leq 2^n\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$

(Lemma 3), and so trying all the possible values of μ in the polynomial-sized set $\{d(1+1/n)^i : 0 \leq i \leq \log_{1+1/n} 2^n\}$ at least one "good" μ. We can then redo the above proof by appropriately modifying some terms by factors of $1-1/n$ or $1+1/n$ in order to satisfy the inequalities that appear. The end result will be that we will have a reduction not from $\mathrm{BDD}_{1/(2\gamma)}$, but from the slightly easier $\mathrm{BDD}_{(1-1/n)^c/(2\gamma)}$ problem for some small constant c. But we can then apply Corollary 2 to obtain the claimed reduction from $\mathrm{BDD}_{1/(2\gamma)}$ to uSVP_γ. $\square$

5 Reducing uSVP to BDD

Theorem 2. *For any polynomially bounded $\gamma(n) = n^{O(1)}$, there is a polynomial time Cook-reduction from uSVP_γ to $\mathrm{BDD}_{1/\gamma}$.*

Proof. Let $\mathbf{B}$ be a uSVP_γ instance, i.e., an n-dimensional lattice such that $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$, and let p be the smallest prime bigger than $\gamma(n)$. (Since $\gamma(n)$ is polynomially bounded, such a prime can be easily found using trial division.) We want to find the shortest nonzero vector in $\mathcal{L}(\mathbf{B})$. We proceed similarly to the reduction from SVP to CVP of Goldreich, Micciancio, Safra and Seifert. (The GMSS reduction corresponds to the special case when $p = 2$.) For any $i = 1, \ldots, n$, we consider the lattice

$$\mathbf{B}^{(i)} = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, p\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n]$$

and invoke the BDD oracle on input $(\mathbf{B}^{(i)}, j \cdot \mathbf{b}_i)$ for $j = 1, \ldots, p-1$. Assume without loss of generality that the oracle always returns a lattice vector. (If the input instance violates the BDD promise, the oracle may simply return $\mathbf{0}$.) Let $\mathbf{v}_{i,j} \in \mathcal{L}(\mathbf{B}^{(i)}) \subset \mathcal{L}(\mathbf{B})$ the lattice vector returned by the oracle on input $(\mathbf{B}^{(i)}, j \cdot \mathbf{b}_i)$, and let

$$\mathbf{w}_{i,j} = \mathbf{v}_{i,j} - j \cdot \mathbf{b}_i.$$

Notice that all vectors $\mathbf{w}_{i,j}$ belong to the lattice $\mathcal{L}(\mathbf{B})$ because $\mathbf{v}_{i,j} \in \mathcal{L}(\mathbf{B})$ and $\mathbf{b}_i \in \mathcal{L}(\mathbf{B})$. The reduction outputs the smallest nonzero vector among the $\mathbf{w}_{i,j}$.

In order to prove the reduction correct, we need to show that at least one of the $\mathbf{w}_{i,j}$ has length $\lambda_1(\mathbf{B})$, so that the reduction outputs a shortest nonzero vector in $\mathcal{L}(\mathbf{B})$. Let $\mathbf{u} = \mathbf{B}\mathbf{x}$ be the shortest nonzero vector in $\mathcal{L}(\mathbf{B})$. Clearly, there must exists an $i \in \{1, \ldots, n\}$ such that p does not divide x_i, because otherwise $\mathbf{u}/p = \mathbf{B}(\mathbf{x}/p) \in \mathcal{L}(\mathbf{B})$ is an even shorter nonzero lattice vector. Fix this i, and let $j = (-x_i \bmod p) \in \{1, \ldots, p-1\}$. We claim that $(\mathbf{B}^{(i)}, j \cdot \mathbf{b}_i)$ is a valid $\mathrm{BDD}_{1/\gamma}$ instance and $\mathrm{dist}(\mathbf{B}^{(i)}, j \cdot \mathbf{b}_i) = \lambda_1(\mathbf{B})$. It will follow that on input $(\mathbf{B}^{(i)}, j \cdot \mathbf{b}_i)$, the BDD oracle returns the lattice vector $\mathbf{v}_{i,j} \in \mathcal{L}(\mathbf{B}^{(i)})$ closest to $j \cdot \mathbf{b}_i$, and

$$\|\mathbf{w}_{i,j}\| = \|\mathbf{v}_{i,j} - j\mathbf{b}_i\| = \mathrm{dist}(\mathbf{B}^{(i)}, j\mathbf{b}_i) = \lambda_1(\mathbf{B}).$$

First, notice that $j\mathbf{b}_i$ is within distance $\|\mathbf{u}\| = \lambda_1(\mathbf{B})$ from $\mathcal{L}(\mathbf{B}^{(i)})$ because

$$\mathbf{u} = \sum_{k=1}^{n} \mathbf{b}_k x_k = \sum_{k \neq i} \mathbf{b}_k x_k + \mathbf{b}_i(x_i + j) - j\mathbf{b}_i$$

and $(x_i + j)$ is a multiple of p. Moreover, $j\mathbf{b}_i \notin \mathcal{L}(\mathbf{B}^{(i)})$ because any vector in $\mathcal{L}(\mathbf{B}^{(i)}) - j\mathbf{b}_i$ uses $\mathbf{b}_i$ a nonzero (modulo p) number of times. Therefore, $\mathrm{dist}(j\mathbf{b}_i, \mathcal{L}(\mathbf{B}^{(i)})) = \lambda_1(\mathbf{B})$. We also need to show that $(\mathbf{B}^{(i)}, j\mathbf{b}_i)$ is a valid $\mathrm{BDD}_{1/\gamma}$ instance, i.e., $\mathrm{dist}(\mathbf{B}^{(i)}, j\mathbf{b}_i) < (1/\gamma)\lambda_1(\mathbf{B}^{(i)})$, or, equivalently, $\lambda_1(\mathbf{B}^{(i)}) > \gamma\lambda_1(\mathbf{B})$. To this end, consider any nonzero vector $\mathbf{y} = \mathbf{B}^{(i)}\mathbf{z}$. If $\mathbf{y}$ is linearly independent from $\mathbf{u}$, then we immediately get $\|\mathbf{y}\| \geq \lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$. So, assume $\mathbf{y} = c\mathbf{u}$ for some $c \in \mathbb{Z}\backslash\{0\}$. Using the definition of $\mathbf{B}^{(i)}$ and the equality $\mathbf{B}^{(i)}\mathbf{z} = c\mathbf{B}\mathbf{x}$, we get $pz_i = cx_i$ (and $z_k = cx_k$ for all $k \neq i$). Since p does not divide x_i (by our choice of i), p must divide c, and $\|\mathbf{y}\| = c\|\mathbf{u}\| \geq p\|\mathbf{u}\| > \gamma\lambda_1(\mathbf{B})$. □

6 Reducing uSVP to GapSVP

Theorem 3. *For any polynomially bounded γ, given an oracle for* GapSVP_γ, *we can solve* uSVP_γ. *Moreover, all calls to the* GapSVP_γ *oracle are of the form* $(\mathbf{B}, d)$ *where* $\lambda_2(\mathbf{B}) > \gamma d$.

Proof. Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ be the basis of a lattice satisfying $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$, and let $\mathbf{u}$ be the (unique) shortest vector in $\mathcal{L}(\mathbf{B})$. Without loss of generality we assume $\mathbf{B}$ is an integer lattice. We show how to use the GapSVP_γ oracle to obtain a lower rank sublattice of $\mathcal{L}(\mathbf{B}')$ that still contains the lattice vector $\mathbf{u}$ of length $\lambda = \lambda_1(\mathbf{B})$. The shortest vector in $\mathcal{L}(\mathbf{B})$ can then be found by iteratively applying this procedure, until the rank of the lattice is reduced to 1, and $\mathbf{B}' = [\pm\mathbf{u}]$.

In fact it is enough to show how to find any full-rank proper sublattice $\mathcal{L}(\mathbf{B}') \subset \mathcal{L}(\mathbf{B})$ still containing $\mathbf{u}$. If we repeat this $t > n(n + \log_2 n)$ times, the result will be a sublattice $\mathbf{S}$ such that $\det(\mathbf{S}) \geq 2^t \det(\mathbf{B})$, because each time we select a sublattice the value of the determinant at least doubles. The dual $\mathbf{D}$ of this sublattice will have determinant $\det(\mathbf{D}) \leq 1/(2^t \det(\mathbf{B}))$, and using the LLL algorithm we can find a dual vector $\mathbf{v} \in \mathcal{L}(\mathbf{D})$ of length

$$\|\mathbf{v}\| \leq 2^n\sqrt{n}\det(\mathbf{D})^{1/n} \leq \frac{\sqrt{n}2^n}{2^{t/n}\det(\mathbf{B})^{1/n}}.$$

By Minkowski's bound we have $\|\mathbf{u}\| \leq \sqrt{n}\det(\mathbf{B})^{1/n}$ and therefore by the Cauchy-Schwarz inequality,

$$|\langle\mathbf{u}, \mathbf{v}\rangle| \leq \|\mathbf{v}\| \cdot \|\mathbf{u}\| \leq n2^{n-t/n} < 1.$$

But $\langle\mathbf{u}, \mathbf{v}\rangle$ is an integer because $\mathbf{u} \in \mathcal{L}(\mathbf{S})$ and $\mathbf{v} \in \mathcal{L}(\mathbf{D})$ and the lattices $\mathcal{L}(\mathbf{S})$ and $\mathcal{L}(\mathbf{D})$ are dual. So, it must be $\langle\mathbf{u}, \mathbf{v}\rangle = 0$, i.e., $\mathbf{u}$ is orthogonal to $\mathbf{v}$. Taking the sublattice of $\mathbf{S}$ orthogonal to $\mathbf{v}$ gives a lower rank sublattice $\mathcal{L}(\mathbf{B}') \subset \mathcal{L}(\mathbf{B})$ still containing $\mathbf{u}$.

So, all we need to do is to show that the GapSVP oracle can be used to find a proper sublattice $\mathcal{L}(\mathbf{B}') \subset \mathcal{L}(\mathbf{B})$ that still contains $\mathbf{u}$. Let p be a prime bigger than γ and consider the sublattices $\mathbf{B}_0 = [p\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k]$ and $\mathbf{B}_c =$

$[\mathbf{b}_1 + c\mathbf{b}_2, p\mathbf{b}_2, \mathbf{b}_3, \ldots, \mathbf{b}_k]$ for $c = 1, \ldots, p$. We claim that there exists a c such that $\mathbf{u} \in \mathcal{L}(\mathbf{B}_c)$. Moreover, for any c, if $\mathbf{u} \notin \mathcal{L}(\mathbf{B}_c)$, then

$$\lambda_1(\mathbf{B}_c) \geq \min(\lambda_2(\mathbf{B}), p\lambda_1(\mathbf{B})) > \gamma\lambda.$$

In other words, the instances $(\mathbf{B}_c, \lambda)$, will always fulfill the promise that either $\lambda_1(\mathbf{B}_c) \leq \lambda$ or $\lambda_1(\mathbf{B}_c) > \gamma\lambda$. So, if we could invoke the GapSVP$_\gamma$ oracle on inputs $(\mathbf{B}_c, \lambda)$ for $c = 0, \ldots, p$, then the oracle would output YES for at least some c, and for any such c we would have $\mathcal{L}(\mathbf{B}_c) \leq \lambda$. However, we cannot make these oracle calls because the value λ is not known, and also because λ might be an irrational number. Both problems can be easily solved by performing a binary search as follows. Compute an approximation d to λ using a lattice approximation algorithm. Say, $\lambda \leq d < 2^n\lambda$. If we invoke the oracle on inputs $(\mathbf{B}_c, d)$, then the oracle will output YES for at least some $c \in \{0, \ldots, p\}$. On the other hand, if we invoke the oracle in inputs $(\mathbf{B}, d/2^n)$, then the oracle will output NO for all c because $d/2^n < \lambda \leq \lambda_1(\mathbf{B}_c)$. Using binary search we can find a $d' < d''$ in $[d/2^n, d)$ such that

– on input $(\mathbf{B}_c, d')$ the oracle outputs NO for all c,
– on input $(\mathbf{B}_c, d'')$ the oracle outputs YES for some c,
– $d'' - d' < 1/(2\gamma^2 d)$.

Notice that the number of iterations performed by the binary search procedure is at most $\log_2(2^n d) + \log_2(2\gamma^2 d) \leq n + 1 + 2\log_2(\gamma d)$ which is polynomial in the input size. From the condition $d'' - d' < 1/(2\gamma^2 d)$, we get that the interval $[(\gamma d')^2, (\gamma d'')^2]$ contains at most one integer because

$$(\gamma d'')^2 - (\gamma d')^2 = \gamma^2(d'' - d')(d'' + d') < 1.$$

Similarly, $(d'')^2 - (d')^2 < 1$ and $[(d')^2, (d'')^2]$ also contains at most one integer. We know that $d' < \lambda$ because the oracle outputs NO on all queries $(\mathbf{B}_c, d')$. Since $\mathbf{B}$ is an integer lattice, λ^2 is an integer. If $[(d')^2, (d'')^2]$ contains no integer value, then it must be $\lambda > d''$, and for all oracle calls $(\mathbf{B}_c, d'')$ that were answered with YES, it must be $\lambda_1(\mathbf{B}_c) \leq \lambda$. On the other hand, if $[(d')^2, (d'')^2]$ contains an integer k, it may or may not be the case that $\lambda = \sqrt{k}$. There are two cases:

– If $(\gamma^2 k, (\gamma d'')^2]$ contains no integer, then for every c, either $\lambda_1(\mathbf{B}_c) \leq d''$ or $\lambda_1(\mathbf{B}_c) > \gamma d''$. So, we can proceed as before, and select any c for which the oracle output YES on input $(\mathbf{B}_c, d'')$.
– If there is an integer $k' \in (\gamma^2 k, (\gamma d'')^2]$, then we select any value $d_0 \in [\sqrt{k}, \sqrt{k'}/\gamma)$, and call the oracle again on input $(\mathbf{B}_c, d_0)$. The oracle will output YES on at least one of these calls, and the corresponding lattice is guaranteed to satisfy $\lambda_1(\mathbf{B}_c) \leq \lambda$.

We will now prove the claim that there exists a $c \in \{0, \ldots, p\}$ such that $\mathbf{u} \in \mathcal{L}(\mathbf{B}_c)$. Let $\mathbf{u} = \mathbf{B}\mathbf{x}$ for some integer vector $\mathbf{x} = (x_1, x_2, \ldots, x_k)^T$. If $p \mid x_1$, then clearly $\mathbf{u}$ is a vector in $\mathcal{L}(\mathbf{B}_0)$. If $p \nmid x_1$, then we will show that $\mathbf{u} \in \mathcal{L}(\mathbf{B}_c)$ for $c = x_2 x_1^{-1} (\bmod\, p)$. Consider the vector $\mathbf{x}' = (x_1, (x_2 - cx_1)/p, x_3, \ldots, x_n)$. Notice that $\mathbf{B}_c\mathbf{x}' = \mathbf{B}\mathbf{x}$ and by our choice of c, $\mathbf{x}$ has all integer coordinates since $x_2 - cx_1 \equiv 0 (\bmod\, p)$. Therefore $\mathbf{u}$ is also a vector in $\mathcal{L}(\mathbf{B}_c)$. □

7 Reducing GapSVP to BDD

In this section we give a reduction from GapSVP to BDD. When combined with the BDD to uSVP reduction from section 4, we obtain the GapSVP to uSVP reduction which proves that the Ajtai-Dwork [2] and the Regev [33] cryptosystems are based on the hardness of the approximate minimum distance problem. As mentioned earlier, the GapSVP to BDD reduction is already implicit in the recent work of Peikert [32]. We repeat it here for completeness and also because in Peikert's work, this reduction is entangled with some extra technicalities that pertain to his main result.

Theorem 4. *For any $\gamma > 2\sqrt{n/\log n}$ there is a polynomial time Cook-reduction from* GapSVP$_\gamma$ *to* BDD$_{\frac{1}{\gamma}\sqrt{n/\log n}}$.

Proof. Let $(\mathbf{B}, d)$ be an instance of GapSVP$_\gamma$. We need to output YES if $\lambda_1(\mathbf{B}) \leq d$ and NO if $\lambda_1(\mathbf{B}) > \gamma d$. In all other instances, any answer will suffice.

We repeat the following procedure $poly(n)$ times. Generate a uniformly random point $\mathbf{s}$ in $\mathcal{B}(0, d\sqrt{n/\log n})$, and let $\mathbf{t} = \mathbf{s} \bmod \mathbf{B}$. Feed the instance $(\mathbf{B}, \mathbf{t})$ to the BDD$_{\frac{1}{\gamma}\sqrt{n/\log n}}$ oracle and receive the answer $\mathbf{v}$. If we ever have the case that $\mathbf{v} \neq \mathbf{t} - \mathbf{s}$, we output YES. On the other hand, if all $poly(n)$ calls to the oracle result in $\mathbf{v}$'s such that $\mathbf{v} = \mathbf{t} - \mathbf{s}$, we output NO.

We will now prove that the reduction is correct. Suppose that $(\mathbf{B}, d)$ is a NO instance of GapSVP$_\gamma$. Then

$$\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) = \mathrm{dist}(\mathbf{s}, \mathcal{L}(\mathbf{B})) \leq d\sqrt{n/\log n} < \frac{\lambda_1(\mathbf{B})}{\gamma}\sqrt{n/\log n},$$

and so $(\mathbf{B}, \mathbf{t})$ is a valid instance of BDD$_{\frac{1}{\gamma}\sqrt{n/\log n}}$. Furthermore, since $\gamma > 2\sqrt{n/\log n}$, the distance of $\mathbf{t}$ from the lattice is less than $\lambda_1(\mathbf{B})/2$, and so there is only one possible lattice vector within distance $\frac{\lambda_1(\mathbf{B})}{\gamma}\sqrt{n/\log n}$ of $\mathbf{t}$. And since the lattice vector $\mathbf{v} = \mathbf{t} - \mathbf{s}$ is at a distance $\|\mathbf{s}\| \leq d\sqrt{n/\log n} < \frac{\lambda_1(\mathbf{B})}{\gamma}\sqrt{n/\log n}$ away from $\mathbf{t}$, it must be the vector that the BDD oracle returns. So, the reduction certainly outputs NO.

Now suppose that that $(\mathbf{B}, d)$ is a YES instance of GapSVP$_\gamma$, which means that $\lambda_1(\mathbf{B}) \leq d$. Let $\mathbf{x}$ be a lattice point whose length is $\lambda_1(\mathbf{B})$. In order for the BDD oracle to successfully fool us into replying NO, he needs to output $\mathbf{v} = \mathbf{t} - \mathbf{s}$ in every round of the protocol. Notice that this is equivalent to the oracle knowing $\mathbf{s}$. But every time we pick an $\mathbf{s}$ and reveal $\mathbf{t} = \mathbf{s} \bmod \mathbf{B}$ to the oracle, by Lemma 4, there is some $1/poly(n)$ probability δ that $\|\mathbf{s} - \mathbf{x}\| \leq d\sqrt{n/\log n}$. And in this case, given $\mathbf{t}$, the oracle cannot know with probability greater than 1/2 whether we randomly generated $\mathbf{s}$ or $\mathbf{s} - \mathbf{x}$ (since both $\mathbf{s} \bmod \mathbf{B}$ and $\mathbf{s} - \mathbf{x} \bmod \mathbf{B}$ equal to $\mathbf{t}$). Therefore for a δ fraction of the $\mathbf{t}$'s that we give him, the oracle cannot guess the exact $\mathbf{s}$ with probability greater than 1/2. And so guessing the $\mathbf{s}$ in all $\mathrm{poly}(n) = n/\delta$ rounds has negligible success probability. Therefore with probability exponentially close to 1, some $\mathbf{v}$ will not equal $\mathbf{t} - \mathbf{s}$ and our algorithm will reply YES. □

8 Reductions for Other ℓ_p Norms

Throughout this work, we have only dealt with the ℓ_2 norm, and we now briefly discuss how our reductions translate to arbitrary ℓ_p norms. The reduction from uSVP to BDD and uSVP to GapSVP in sections 5 and 6 don't rely on any specific properties of the ℓ_2 norm and so the reductions go through for other norms with only very slight modifications. In the reduction from BDD to uSVP in section 4, we repeatedly used the definition of the ℓ_2 norm, and so the reductions do not go straight through. Nevertheless, a simple modification of the proof which involves appropriately changing the equalities and inequalities to correspond with the definitions of the ℓ_p norm of interest, results in a reduction from $\text{BDD}_{1/(2\gamma)}$ to uSVP_γ just as for the ℓ_2 norm.

The only reduction that becomes weaker for ℓ_p norms where $p \neq 2$ is the reduction from GapSVP to BDD in section 7. The $\sqrt{n/\log n}$ factor loss in the reduction for the ℓ_2 norm is directly tied to the fact that spheres that are a distance of d apart must have radii of at least $d\sqrt{n/\log n}$ in order for their intersecting volume to be a non-negligible fraction of their total volume (Lemma 4). On the other hand, in ℓ_p norms for $p \neq 2$, the radii of the spheres have to be larger for their intersection to be non-negligible. It's not hard to see that for the ℓ_1 and ℓ_∞ norms, the radii need to be at least $dn/\log n$, and it is shown in [9] that this suffices for all other ℓ_p norms as well (although it is not a tight bound when $1 < p < \infty$). So essentially using an analogue of Lemma 4 for other ℓ_p norms, we can obtain a reduction from GapSVP_γ to $\text{BDD}_{\frac{1}{\gamma}n/logn}$ for any $\gamma > 2n/\log n$.

Acknowledgements. We thank the anonymous referees for very useful comments.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: STOC, pp. 99–108 (1996)
2. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: STOC (1997); An improved version is described in ECCC 2007 (2007)
3. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: STOC, pp. 601–610 (2001)
4. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986)
5. Cai, J.Y.: A relation of primal-dual lattices and the complexity of shortest lattice vector problem. Theor. Comput. Sci. 207(1), 105–116 (1998)
6. Cai, J.Y.: On the average-case hardness of CVP. In: FOCS, pp. 308–317 (2001)
7. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
8. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices, and new cryptographic constructions. In: STOC (2008)
9. Goldreich, O., Goldwasser, S.: On the limits of nonapproximability of lattice problems. J. Comput. Syst. Sci. 60(3), 540–563 (2000)

10. Goldreich, O., Goldwasser, S., Halevi, S.: Collision-free hashing from lattice problems. In: Electronic Colloquium on Computational Complexity (ECCC) (1996)
11. Goldreich, O., Goldwasser, S., Halevi, S.: Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 105–111. Springer, Heidelberg (1997)
12. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)
13. Goldreich, O., Micciancio, D., Safra, S., Seifert, J.P.: Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. Information Processing Letters 71(2), 55–61 (1999)
14. Haviv, I., Regev, O.: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In: STOC, pp. 469–477 (2007)
15. 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)
16. Impagliazzo, R.: A personal view of average-case complexity. In: Structure in Complexity Theory Conference, pp. 134–147 (1995)
17. Kannan, R.: Algorithmic geometry of numbers. Annual Review of Computer Science 2, 231–267 (1987)
18. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer, Heidelberg (2008)
19. Khot, S.: Hardness of approximating the shortest vector problem in lattices. In: FOCS, pp. 126–135 (2004)
20. Klivans, A., Sherstov, A.: Cryptographic hardness for learning intersections of halfspaces. In: FOCS, pp. 553–562 (2006)
21. Kumar, R., Sivakumar, D.: On the unique shortest lattice vector problem. Theor. Comput. Sci. 255(1-2), 641–648 (2001)
22. Lagarias, J.C., Odlyzko, A.M.: Solving low density subset sum problems. Journal of the ACM 32, 229–246 (1985)
23. Lenstra, A.K., Lenstra Jr., H.W., Lovasz, L.: Factoring polynomials with rational coefficients. Mathematische Annalen (261), 513–534 (1982)
24. Liu, Y.-K., Lyubashevsky, V., Micciancio, D.: On bounded distance decoding for general lattices. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 450–461. Springer, Heidelberg (2006)
25. Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 162–179. Springer, Heidelberg (2008)
26. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
27. Micciancio, D.: Efficient reductions among lattice problems. In: SODA, pp. 84–93 (2008)
28. Micciancio, D., Goldwasser, S.: Complexity Of Lattice Problems: A Cryptographic Perspective. Kluwer Academic Publishers, Dordrecht (2002)
29. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. on Computing 37(1), 267–302 (2007)

30. Micciancio, D., Vadhan, S.: Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 282–298. Springer, Heidelberg (2003)
31. Nguyen, P.Q.: Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto 1997. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 288–304. Springer, Heidelberg (1999)
32. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC (2009)
33. Regev, O.: New lattice-based cryptographic constructions. J. ACM 51(6), 899–942 (2004)
34. Regev, O.: Quantum computation and lattice problems. SIAM J. Comput. 33(3), 738–760 (2004)
35. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC (2005)

Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems

Benny Applebaum[1,⋆], David Cash[2], Chris Peikert[3,⋆⋆], and Amit Sahai[4,⋆⋆⋆]

[1] Princeton University
[2] Georgia Institute of Technology
[3] SRI International
[4] UCLA

Abstract. The well-studied task of *learning a linear function with errors* is a seemingly hard problem and the basis for several cryptographic schemes. Here we demonstrate additional applications that enjoy strong security properties and a high level of efficiency. Namely, we construct:

1. Public-key and symmetric-key cryptosystems that provide security for *key-dependent messages* and enjoy *circular security*. Our schemes are highly efficient: in both cases the ciphertext is only a constant factor larger than the plaintext, and the cost of encryption and decryption is only $n \cdot \text{polylog}(n)$ bit operations per message symbol in the public-key case, and $\text{polylog}(n)$ bit operations in the symmetric-case.
2. Two efficient pseudorandom objects: a "weak randomized pseudorandom function" — a relaxation of standard PRF — that can be computed obliviously via a simple protocol, and a length-doubling pseudorandom generator that can be computed by a circuit of $n \cdot \text{polylog}(n)$ size. The complexity of our pseudorandom generator almost matches the complexity of the fastest known construction (Applebaum *et al.*, RANDOM 2006), which runs in linear time at the expense of relying on a nonstandard intractability assumption.

Our constructions and security proofs are simple and natural, and involve new techniques that may be of independent interest. In addition, by combining our constructions with prior ones, we get fast implementations of several other primitives and protocols.

Keywords: Encryption, Key-dependent message security, Learning problems, Lattice-based cryptography.

⋆ This material is based upon work supported by the National Science Foundation under Grants CNS-0627526, CCF-0426582 and CCF-0832797. 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.

⋆⋆ This material is based upon work supported by the National Science Foundation under Grants CNS-0716786 and CNS-0749931.

⋆⋆⋆ Research supported in part from NSF grants 0716389, 0627781, 0830803, a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Alfred P. Sloan Foundation Fellowship, and an Okawa Foundation Research Grant.

1 Introduction

The problem of "learning a linear function with errors" (LWE) has found many interesting cryptographic and complexity-theoretic applications in the last few years (see [32,36,46,37,45,25], to name a few). Informally, the LWE problem, for a dimension n and modulus q, is to recover a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ given arbitrarily many "noisy random inner products" $(\mathbf{a}_i, b_i \approx \langle \mathbf{a}_i, \mathbf{s} \rangle) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where the $\mathbf{a}_i \in \mathbb{Z}_q^n$ are uniform and independent. The "learning parity with noise" problem (LPN) is the special case where $q = 2$. These problems have been studied extensively in several works, and their known best algorithms require $2^{O(n \log q / \log n)}$ time and space [11].

Much evidence suggests that no efficient algorithm can solve LWE/LPN with better than negligible probability, even using quantum computation. In particular, the LPN problem can be formulated as the famous problem of decoding a random binary linear code, and therefore a successful attack would imply a major breakthrough in coding theory. The LPN problem also occupies a central position in learning theory: an efficient algorithm for it could be used to learn several important concept classes, including 2-DNF formulas, juntas, and *any* function with a sparse Fourier spectrum [21].

For the case of the LWE, hardness is supported by a remarkable connection to *worst-case* lattice problems. Regev [46] showed that solving LWE (for certain Gaussian-like error distributions) is as hard as *quantumly* solving some apparently intractable lattice problems, such as the approximate shortest vector problem GapSVP. Recently, Peikert [43] also gave a *classical* reduction from GapSVP (and variants) to LWE.

The LWE/LPN problems provide an interesting combination of two useful properties: *efficiency*, i.e., instances of the problem can be generated by "cheap" operations such as (modular) addition and multiplication (or even simple *bit operations* in the case of LPN), and *simple algebraic structure*, i.e., the noisy inner products are computed by an "almost linear" function. Indeed, previous works relied on these properties to obtain cryptography with low complexity [32,4,35,6], and to derive desirable cryptographic features such as random self-reducibility (with respect to the choice of $\mathbf{a}$) and pseudorandomness [10]. Interestingly, other problems that provide the latter feature, such as those from number theory, typically require relatively expensive computational operations such as exponentiation over large groups.

1.1 Our Results

In this paper, we further exploit the properties of LWE/LPN to obtain new cryptographic constructions that are both efficient and enjoy desirable security properties.

Circular-secure encryption schemes. One of our main applications is the construction of efficient encryption schemes (in both the symmetric- and public-key settings) that achieve security against certain *key-dependent message* (KDM) attacks [9]; that is, they remain secure even when the adversary is allowed to obtain encryptions of messages that depend on the secret keys themselves, via any affine function of the adversary's choice. Moreover, our schemes are "circular secure" [15], that is, they remain secure even in the presence of key "cycles" or even "cliques," where any user's secret key may be encrypted under any user's public key. Such usage arises in key-management systems, in anonymous credential systems [15], and in the context of "axiomatic security" [1] (See [13] for a detailed discussion).

In the last few years, the notions of KDM and circular security have been studied extensively [29,8,14,7,31,28]. Without resorting to the use of random oracles, constructing a circular-secure encryption scheme (either in the private-key or public-key setting) was a long-standing open problem. This question was recently resolved by Boneh *et al.* [13], who constructed such a *public-key* encryption scheme based on the DDH assumption. Their construction relies on a clever use of the homomorphic properties of the DDH problem. However, exploiting these properties incurs a large overhead in both computation and communication. In contrast, our approach yields very *natural* encryption schemes that have significant efficiency advantages over the prior scheme of [13].

The contrast is clearest when we compare the current "cost" of achieving security for key-dependent messages against the cost of achieving ordinary semantic security, for a given computational intractability assumption. Comparing the scheme of [13] to other semantically secure encryption schemes based on the DDH problem, the cost is dramatic: While standard encryption schemes like ElGamal can encrypt their key, which is about $k = \log |\mathbb{G}|$ bits (where $\mathbb{G}$ is the underlying group of the DDH problem), using a single exponentiation and one group element of overhead in the ciphertext, the scheme given in [13] requires about k exponentiations and group elements of ciphertext overheard *per bit of the secret key.* Encrypting key-independent messages is about k times more efficient, but it still incurs a factor k loss over standard ElGamal. In contrast, our constructions are essentially *as efficient* as prior semantically secure schemes based on essentially the same hardness assumptions.

Specifically, our *public-key* schemes are variants of Regev's LWE-based scheme and the more-efficient amortized version of Peikert, Vaikuntanathan, and Waters [44], with several non-trivial modifications to facilitate the proof of security for key-dependent messages. The most efficient version takes only $\tilde{O}(n)$ amortized time per message symbol for both encryption and decryption, and the ciphertext is only a constant factor larger than the plaintext.

Our *symmetric-key* cryptosystem is based on the LPN problem. Its ciphertexts are only a constant factor larger than the plaintexts, and both encryption and decryption can be performed by Boolean circuits of quasi-linear size (in the message length), which is almost optimal even for standard CPA-security. The scheme is a close variant of the LPN-based encryption scheme of Gilbert *et al.* [26], which was proved secure only in the standard sense (i.e., without key-dependent messages), and did not achieve quasi-linear time efficiency. The scheme was discovered independently by the first author and by Dodis *et al.* [19], who proved security in the presence of key-leakage under a stronger version of the LPN assumption. We stress that key-leakage security is incomparable to the notions studied here.

Fast Pseudorandom Objects

Pseudorandom generator. Based on the hardness of LPN, we construct a pseudorandom generator (PRG) that doubles its input length and can be computed by a Boolean circuit of size $\tilde{O}(n)$ (i.e., quasilinear size). This is considerably faster than previous constructions of linear-stretch PRGs (e.g., [18,17,24]), which suffer from *polynomial* overhead. (Similar limitations also hold for previous coding-based constructions [10,22].) To the best of our knowledge, the only exception is the construction of [5] which is computable by linear-size ($\mathbf{NC^0}$) circuits. This construction is based on a plausible, yet non-standard, assumption of Alekhnovich [3]. Roughly speaking, that assumption says that a noisy random codeword of a code with *sparse* generating matrix is pseudorandom. This assumption is relatively new and, while seemingly reasonable, it has not been widely studied yet. Moreover, unlike our LPN-based assumption, Alekhnovich's assumption posits *pseudorandomness* rather than just *one-wayness*, which is in general a stronger notion.

Application. Typical cryptographic functions introduce a multiplicative computational overhead that grows with the desired level of security. Recently, Ishai *et al.* [34] showed that many cryptographic tasks can be implemented while incurring only a constant computational overhead compared to insecure implementations of the same tasks.[1] These results were based on the PRG construction of [5], and hence on non-standard intractability assumptions.

By plugging our PRG into the reductions of [34], we get implementations with polylogarithmic overhead for several primitives such as commitment schemes, symmetric encryption schemes, and public-key encryption schemes (under the assumption that the latter exist). This provides an interesting alternative to the original suggestion of [34], as it relies on a standard assumption and still gives a considerable improvement over typical (non-IKOS) schemes.[2] We view this result as an important support for the possibility of cryptography with low overhead.

Randomized weak pseudorandom function. We also obtain a simple construction of randomized weak pseudorandom function family (RWPRFs). This primitive

[1] We make the usual security requirement that the advantage of any polynomial-time attacker must be negligible in the input length.

[2] A trivial construction of primitives with polylogarithmic security can be achieved by starting from an exponentially strong primitive (e.g., PRG) and applying it separately on input blocks of polylogarithmic size. This construction results in primitives with weak (quasi-polynomial) security. In contrast, our construction starts from a weaker assumption (in particular, it does not require exponential hardness) and it results in a primitive whose security is essentially the same as the security of the assumption (up to standard polynomial loss).

relaxes the standard notion of pseudorandom function family [27] in two ways: it provides security only when the function is evaluated on randomly chosen points, and it uses secret internal randomness. To make this notion nontrivial we require an efficient "equality-tester" that verifies whether different invocations of the PRF (with independent internal randomness) correspond to the same preimage. While this primitive is considerably weaker than PRFs, we argue that in some scenarios RWPRFs can be used instead of standard PRFs. Moreover, the use of internal randomness provably admits more efficient constructions.[3]

Our construction has several interesting syntactic properties: it is injective and symmetric (one can replace the roles of the argument and the key without violating security). Moreover, we describe a simple constant-round protocol for obliviously evaluating the function. Such a protocol allows two parties, one holding a point x and another holding a key k, to evaluate the function $f_k(x)$ without learning each other's inputs. Pseudorandom functions that allow oblivious evaluation (OPRFs) were recently shown to be a useful cryptographic tool [23,30]. An oblivious RWPRF can replace an OPRF in some settings (despite its weaker cryptographic properties), and hence our construction provides an alternative to the relatively small number of existing schemes (see [23] and references within).

Practical efficiency vs. asymptotic efficiency. In this paper, we treat efficiency in *asymptotic* terms. Still, we believe that some of our results may turn to be useful in practice as well. Indeed, our LPN-based constructions mainly rely on addition and multiplication of large binary matrices. These operations can be performed very fast in practice [12,2] even if one does not employ the asymptotically-fast algorithms used in our analysis (e.g., for matrix multiplication), which might not be applicable in practice. In particular, as in the case of the HB protocol [32,35], our schemes (or variants of them) might turn to be useful for hardware implementation by computationally-weak devices. We leave this direction for future study.

1.2 Techniques

Our LWE-based public key construction involves a few techniques that may be of independent interest and application.

In the LWE-based cryptosystems of [46,44], the secret key is a vector $\mathbf{s} \in \mathbb{Z}_q^n$ chosen uniformly at random, while the message space is $\mathbb{Z}_p$ for some $p \ll q$. An important idea in the work of Boneh *et al.* [13] is the ability to generate, given only a public key, a ciphertext that decrypts to a message related to $\mathbf{s}$. Because decryption in the LWE-based schemes of [46,44] is essentially a linear operation, it is easy to generate ciphertexts that are somehow related to $\mathbf{s}$. However, because

[3] For example, it can be shown that PRFs cannot be computed by constant-depth circuits with unbounded fan-in AND and XOR gates [38]. In contrast, our construction can be computed by such circuits of depth 2. Moreover, one can show that, under plausible assumptions, RWPRFs can be constructed even in weaker classes such as $\mathbf{NC^0}$.

the entries of $\mathbf{s}$ are taken modulo $q \gg p$, it is unclear how to "fit" the entries into the message space.

We address this issue by instead drawing the entries of the secret key $\mathbf{s}$ from the very same (Gaussian) *error distribution* as in the underlying LWE problem. For a sufficiently "narrow" error distribution, each entry of $\mathbf{s}$ can take on at most p different values (with overwhelming probability), allowing the entire entry to fit unambiguously into the message space. Moreover, this change *does not affect the hardness of the* LWE *problem*: we show a simple, tight reduction from the standard LWE problem to the variant just described. Abstractly, the reduction may be viewed as putting the LWE distribution into *Hermite normal form* (HNF); interestingly, the HNF was also used by Micciancio [39] and Micciancio and Regev [41] as a way to improve the *efficiency* of lattice-based cryptosystems.

The second important technique relates to the faithful simulation of key-dependent messages. We modify the encryption algorithms of [46,44] to ensure that ciphertexts *themselves* have a "nice" distribution that supports the desired homomorphisms. Essentially, our encryption algorithms apply Regev's worst-case to average-case reduction (from lattices to LWE) to the (already random) public key itself; we also generalize Regev's analysis to deal with the amortized system of [44]. In addition, to support the homomorphisms we need to rely on LWE with a *prime power* modulus $q = p^e$, where p is the size of the message space. Fortunately, a hybrid-argument extension of the usual pseudorandomness proof [10,46] for LWE also works for prime power moduli, as long as the error distribution is sufficiently "narrow."

A final interesting technique concerns a more general attack involving key cycles/cliques, where every user's secret key may be encrypted under every user's public key. Simulating such a scenario seems to require knowing a relation between every pair of (unknown and independent) secret keys. Conveniently, the above-described transformation for LWE (allowing the secret $\mathbf{s}$ to be drawn from the error distribution) can also be used to produce many independent keys, and happens to produce the desired linear relations among them as a side effect!

2 Preliminaries

For a probability distribution X over a domain D, let X^n denote its n-fold product distribution over D^n. The uniform distribution over a finite domain D is denoted $U(D)$. We write U_n to denote the special case of the uniform distribution over $\{0,1\}^n$ and (by abuse of notation) the uniform distribution over $\mathbb{Z}_2^n$. Let Ber_ε denote the Bernoulli distribution over $\{0,1\}$ that is 1 with probability ε and 0 with probability $1-\varepsilon$.

We write $\mathrm{negl}(n)$ to denote an arbitrary *negligible* function, i.e., one that vanishes faster than the inverse of any polynomial. We say that that a probability is *overwhelming* if it $1-\mathrm{negl}(n)$.

The *statistical distance* between two distributions X and Y over a countable domain D (or two random variables having those distributions) is defined as $\varDelta(A,B) = \max_{A \subseteq D} |f_X(A) - f_Y(A)|$. We write $X \equiv Y$ if the two random variable are identically distributed. We say that two ensembles $\{X_n\}$ and $\{Y_n\}$ of distributions indexed by n are *statistically indistinguishable* if $\Delta(X_n, Y_n) = \text{negl}(n)$. The ensembles are *computationally indistinguishable* if for every probabilistic polynomial-time adversary $\mathcal{A}$, the distinguishing advantage $|\Pr[\mathcal{A}(X_n) = 1] - \Pr[\mathcal{A}(Y_n) = 1]| = \text{negl}(n)$. A distribution ensemble $\{X_n\}_{n \in \mathbb{N}}$ is *pseudorandom* if X_n is computationally indistinguishable from $U(D_n)$ where D_n is the domain of X_n (which is usually clear from the context).

2.1 Noisy Learning Problems

We recall the learning with error (LWE), due to Regev [46], for which learning parity with noise (LPN) is a special case.

For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, define $A_{\mathbf{s},\chi}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, an error term $x \leftarrow \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + x)$.

Definition 1. *For an integer function $q = q(n)$ and an error distribution χ over $\mathbb{Z}_q$, the* learning with errors *problem* $\mathsf{LWE}_{q,\chi}$ *in n dimensions is defined as follows: given access to an oracle that produces independent samples from $A_{\mathbf{s},\chi}$ for some arbitrary $\mathbf{s} \in \mathbb{Z}_q^n$, output $\mathbf{s}$ with noticeable probability, e.g., $1/2$, over all the randomness of the oracle and the algorithm.*

The learning parity with noise *problem* LPN_ε *is the special case of* $\mathsf{LWE}_{q,\chi}$ *for $q = 2$ and $\chi = \mathsf{Ber}_\varepsilon$.*

We say that $\mathsf{LWE}_{q,\chi}$ is *hard* (or intractable) for a class of adversaries (by default, probabilistic poly(n)-time algorithms) if there does not exist an algorithm in the class that can solve it for infinitely many n.

Note that LWE as defined above is a "worst-case" style of problem in that the value of $\mathbf{s} \in \mathbb{Z}_q^n$ is arbitrary, not random as is typical in cryptography. This is not too important of a distinction, because LWE is amenable to randomized self-reduction and amplification techniques [10,46]. In particular, here we give a reduction from the form of the problem in Definition 1 to an *average-case decision problem*, for *prime power moduli* and "narrow enough" error distributions. In other words, under the hypotheses of the lemma, the LWE distribution is pseudorandom if the search problem is hard.

Lemma 1. *Let $q = p^e$ be a prime power with $p = \text{poly}(n)$, and let χ be a distribution over $\mathbb{Z}_q$ that produces an element in $\{-\frac{p-1}{2}, \ldots, \frac{p-1}{2}\} \subset \mathbb{Z}_q$ with overwhelming probability. There is a probabilistic polynomial-time reduction from solving* $\mathsf{LWE}_{q,\chi}$ *to distinguishing (with non-negligible advantage) between $A_{\mathbf{s},\chi}$ for uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$ and the uniform distribution $U = U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.*

Proof (sketch). The proof is a simple extension of prior ones for prime moduli (see, e.g., [46, Lemma 4.2]), therefore we sketch only the new elements. The idea is to use a distinguisher to recover the *least significant* digit (in base p) of each entry of $\mathbf{s}$, after which the error distribution can be made narrow enough to solve for all the remaining digits of $\mathbf{s}$ via rounding and linear algebra. Due to space limitations, the entire proof is deferred to the full version.

For $i = 0, \dots, e$, define the hybrid distribution $A^i_{\mathbf{s},\chi}$ that is obtained by drawing a sample $(\mathbf{a}, b)$ from $A_{\mathbf{s},\chi}$ and outputting $(\mathbf{a}, b + p^i \cdot r) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ for a uniformly random $r \in \mathbb{Z}_q$ (freshly chosen for each sample). By a hybrid argument and standard amplification techniques, we can use an algorithm D that distinguishes between $A_{\mathbf{s},\chi}$ and U to solve for $\mathbf{s}' = \mathbf{s} \bmod p$. Having done so, we can then transform $A_{\mathbf{s},\chi}$ into $A_{p\cdot\mathbf{t},\chi}$, where $p \cdot \mathbf{t} = \mathbf{s} - \mathbf{s}' \in \mathbb{Z}_q^n$. A sample from the latter distribution is of the form $(\mathbf{a}, b = p \cdot \langle \mathbf{a}, \mathbf{t} \rangle + x)$ for $x \leftarrow \chi$; because $x \in \{-\frac{p-1}{2}, \dots, \frac{p-1}{2}\}$ with overwhelming probability, we may round b to the nearest multiple of p and learn the value of $\langle \mathbf{a}, \mathbf{t} \rangle \bmod p$ exactly. With enough samples of this form, we may then solve for $\mathbf{t}$ by linear algebra.

We are interested in error distributions χ over $\mathbb{Z}_q$ that are derived from Gaussians. For any $r > 0$, define the one-dimensional Gaussian probability distribution by its density function $D_r(x) = \exp(-\pi(x/r)^2)/r$. For $\alpha > 0$, define $\bar{\Psi}_\alpha$ to be the distribution on $\mathbb{Z}_q$ obtained by drawing $y \leftarrow D_\alpha$ and outputting $\lfloor q \cdot y \rceil \bmod q$. Regev [46] demonstrated strong evidence for the hardness of the LWE problem with such a Gaussian error distribution, by giving a *quantum* reduction from approximating well-studied lattice problems to within $\tilde{O}(n/\alpha)$ factors in the *worst case* to solving $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$, when (say) $\alpha \cdot q \geq n$. Recently, Peikert [43] also gave a related *classical* reduction for similar parameters.

For our public-key encryption algorithms, we also need the *discrete Gaussian* distribution $D_{\mathbb{Z}^m,r}$ over the integer lattice $\mathbb{Z}^m$, which assigns probability proportional to $\prod_{i\in[m]} D_r(x_i)$ to each $\mathbf{x} \in \mathbb{Z}^m$. It is possible to sample efficiently from $D_{\mathbb{Z}^m,r}$ for any $r > 0$ [25].

2.2 Key-Dependent Message Security

We now define key-dependent message security for encryption, following the presentation of Boneh *et al.* [13], which generalizes the definition of Black *et al.* [9]. In this definition, an adversary plays a game with a challenger that answers encryption queries for functions of the users' secret keys. The adversary is restricted to queries for functions from a certain family, which we will denote $\mathcal{F} \subset \{f \mid f : \mathcal{K}^\ell \to \mathcal{M}\}$, where $\mathcal{K}$ and $\mathcal{M}$ are the keyspace and message space of the encryption scheme. Strictly speaking, $\mathcal{F}$ is a family of sets of functions parameterized by the security parameter n and the number of users ℓ.

Let us fix a public-key encryption scheme, and let $\mathcal{A}$ be an adversary. We will write $\mathsf{Enc}(pk, m)$ to denote encrypting message m under public key pk. The game proceeds as follows:

1. The challenger chooses a bit $b \leftarrow \{0, 1\}$. It also chooses $(pk_1, sk_1), \dots, (pk_\ell, sk_\ell)$ by running the scheme's key generation algorithm ℓ times. It gives $pk_1, \dots, pk_\ell$ to the adversary.
2. $\mathcal{A}$ makes encryption queries of the form (i, f), where $1 \leq i \leq \ell$ and $f \in \mathcal{F}$. To process a query, if $b = 0$, the challenger computes $m \leftarrow f(sk_1, \dots, sk_\ell)$

and $c \leftarrow \mathsf{Enc}(pk_i, m)$. If $b = 1$ it instead sets $c \leftarrow \mathsf{Enc}(pk_i, 0^{|m|})$. It returns c to $\mathcal{A}$.

3. $\mathcal{A}$ attempts to guess b and outputs $\hat{b} \in \{0, 1\}$.

The scheme is *KDM-CPA secure with respect to* $\mathcal{F}$ if for every efficient adversary $\mathcal{A}$, the probability of guessing b is at most $\frac{1}{2} + \text{negl}(n)$ for some negligible function $\text{negl}(\cdot)$.

We can define KDM-CPA security for symmetric key encryption similarly: in phase one, the challenger generates secret keys and gives the adversary nothing, and in phase two it uses the secret keys to encrypt (and as input to f). Everything else is exactly the same. Finally, the definition of *CCA-KDM security* is similar except that the adversary has also an oracle access to the decryption function $\mathsf{Dec}(k, \cdot)$ (but cannot query this oracle on any output given to him by the encryption oracle).

If all constant functions (that is, functions f_m such that $f_m(k_1, \ldots, k_\ell) = m$ for some $m \in \mathcal{M}$) are contained in $\mathcal{F}$, then security with respect to $\mathcal{F}$ implies standard CPA security. If the projection functions (f_j such that $f_j(k_1, \ldots, k_\ell) = k_j$ for some j) are contained in $\mathcal{F}$, then security with respect to $\mathcal{F}$ implies (and is actually stronger than) circular security.

3 Public-Key Encryption

In this section we design a public-key cryptosystem based on the $\mathsf{LWE}_{q,\chi}$ problem, where as usual, the error distribution χ is the discretized Gaussian $\bar{\Psi}_\alpha$ for parameter $\alpha = \alpha(n) \in (0, 1)$, and the modulus q is chosen to satisfy various constraints.

3.1 A Generic Transformation

We start with a useful transformation that reduces the LWE problem to one in which the *secret itself* is chosen from the error distribution χ, essentially putting the LWE distribution into "Hermite normal form."

Lemma 2. *Let $q = p^e$ be a prime power. There is a deterministic polynomial-time transformation T that, for arbitrary $\mathbf{s} \in \mathbb{Z}_q^n$ and error distribution χ, maps $A_{\mathbf{s},\chi}$ to $A_{\bar{\mathbf{x}},\chi}$ where $\bar{\mathbf{x}} \leftarrow \chi^n$, and maps $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ to itself. The transformation also produces an invertible square matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times n}$ and $\bar{\mathbf{b}} \in \mathbb{Z}_q^n$ that, when mapping $A_{\mathbf{s},\chi}$ to $A_{\bar{\mathbf{x}},\chi}$, satisfy $\bar{\mathbf{x}} = -\bar{\mathbf{A}}^T \mathbf{s} + \bar{\mathbf{b}}$.*

Proof. The transformation T is given access to some distribution D over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ (where D may be either $A_{\mathbf{s},\chi}$ or $U = U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$), and proceeds in two stages.

In the first stage, T performs some initial processing to obtain $\bar{\mathbf{A}}, \bar{\mathbf{b}}$. It does this by drawing several pairs $(\mathbf{a}, b)$ from D, and keeping certain of them until it has accumulated a set of n pairs $\{(\bar{\mathbf{a}}_i, \bar{b}_i)\}$ that will make up $\bar{\mathbf{A}}, \bar{\mathbf{b}}$ in the natural way. With each new sample $(\mathbf{a}, b)$, T checks whether $\mathbf{a}$ is linearly independent modulo q of all those $\bar{\mathbf{a}}_i$ that have been kept so far; if so, $(\mathbf{a}, b)$ is kept, otherwise it is discarded. Note that the probability of keeping a particular sample is at least $\varphi(q)/q \geq 1/2$ (where φ denotes the Euler totient function), so with high probability, T accumulates the required n samples after drawing $O(n^2)$ samples from D. Now by construction, $\bar{\mathbf{A}}$ is invertible modulo q. Also observe that each sample is kept or discarded based only on its $\mathbf{a}$ component, so when $D = A_{\mathbf{s},\chi}$, we have $\bar{\mathbf{b}} = \bar{\mathbf{A}}^T\mathbf{s} + \bar{\mathbf{x}}$ where $\bar{\mathbf{x}}$ is drawn from χ^n.

The second stage actually transforms (fresh) samples from D into samples from a possibly different distribution. Given a draw $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ from D, T outputs $(\mathbf{a}', b') \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where

$$\mathbf{a}' = -\bar{\mathbf{A}}^{-1}\mathbf{a} \quad \text{and} \quad b' = b + \langle \mathbf{a}', \bar{\mathbf{b}} \rangle.$$

Observe that because $\bar{\mathbf{A}}$ is invertible modulo q and $\mathbf{a} \in \mathbb{Z}_q^n$ is uniform, $\mathbf{a}' \in \mathbb{Z}_q^n$ is uniform as well. We now consider the two cases for D. If $D = U$, then $(\mathbf{a}', b')$ is also distributed according to U, because $b \in \mathbb{Z}_q$ is uniform and independent of $\mathbf{a}$. If $D = A_{\mathbf{s},\chi}$, then $b = \langle \mathbf{a}, \mathbf{s} \rangle + x$ for some $x \leftarrow \chi$, so we have

$$b' = \langle \mathbf{a}, \mathbf{s} \rangle + x - \langle \bar{\mathbf{A}}^{-1}\mathbf{a}, \bar{\mathbf{A}}^T\mathbf{s} \rangle + \langle \mathbf{a}', \bar{\mathbf{x}} \rangle = \langle \mathbf{a}', \bar{\mathbf{x}} \rangle + x.$$

Therefore, $(\mathbf{a}', b')$ is distributed according to $A_{\bar{\mathbf{x}},\chi}$, as desired.

3.2 The Cryptosystem

We now define a KDM-secure cryptosystem based on the LWE problem. For technical reasons, our construction uses a *prime power* modulus $q = p^2$ of a certain size, with messages taken over $\mathbb{Z}_p$. (Other choices of $q = p^e$ are possible, but $q = p^2$ seems to correspond to the mildest underlying assumption.) Note that any element $v \in \mathbb{Z}_q$ may be written as $v = (v_1, v_0) \in \mathbb{Z}_p \times \mathbb{Z}_p$, where v_1 and v_0 are the most and least significant digits in the base-p representation of v, respectively, with the digits chosen from the set of residues $\{-\frac{p-1}{2}, \ldots, \frac{p-1}{2}\}$. Recall that by Lemma 1, the LWE distribution $A_{\mathbf{s},\chi}$ (for uniform $\mathbf{s} \in \mathbb{Z}_q^n$) is pseudorandom if the search problem $\mathsf{LWE}_{q,\chi}$ is hard, and if the error distribution χ is concentrated on $\{0\} \times \mathbb{Z}_p$ (which will be the case in our system, by design).

For simplicity, we start with a scheme that encrypts a single element of $\mathbb{Z}_p$ at a time, later extending it to an amortized version in Section 3.4. Our scheme is very similar to Regev's cryptosystem [46], with two main differences. First, the entries of the secret key $\mathbf{s} \in \mathbb{Z}_q^n$ are chosen from the (narrow) *error distribution* χ (rather than uniformly), so that they may be represented unambiguously as elements of the message space $\mathbb{Z}_p$ (see Lemma 3); this is secure due to Lemma 2. Second, we modify the encryption algorithm so that it induces a 'nice' distribution over ciphertexts (see Lemma 4). Specifically, the encryption algorithm chooses a random vector $\mathbf{r} \in \mathbb{Z}^m$ from a discrete Gaussian distribution (rather than from $\{0,1\}^m$), and adds a small extra term e to 'smooth out' the ciphertext distribution. These steps may be seen as applying Regev's main worst-case to average-case reduction [46] to the (already random) public key.

Construction 1. *The construction is parametrized by $q = p^2$ for some prime p, and an error parameter α; we instantiate these parameters below. Let $\chi = \bar{\Psi}_\alpha$, the discretized Gaussian over $\mathbb{Z}_q$.*

- Key generation: *The secret key is $\mathbf{s} \leftarrow \chi^n$. The public key is $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, which is made up of $m \geq 2(n+1)\lg q$ draws $(\mathbf{a}_i, b_i)$ from $A_{\mathbf{s},\chi}$. That is, $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{x}$ for independent $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{x} \leftarrow \chi^m$.*
- Encryption: *Before specifying the encryption algorithm, we define a distribution $E_{\mathbf{A},\mathbf{b}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, which has parameters $r = \omega(\sqrt{\log m})$ and $r' = r \cdot \sqrt{m} \cdot (\alpha + \frac{1}{2q})$. The distribution is obtained by choosing $\mathbf{r} \leftarrow D_{\mathbb{Z}^m,r}$ and $e \leftarrow \bar{\Psi}_{r'}$ and outputting*

$$(\mathbf{A}\mathbf{r}, \langle \mathbf{r}, \mathbf{b}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

  To encrypt a message $z \in \mathbb{Z}_p$ given the public key $(\mathbf{A}, \mathbf{b})$, draw a sample $(\mathbf{u}, v)$ from $E_{\mathbf{A},\mathbf{b}}$ and output the ciphertext $(\mathbf{u}, c = v + z \cdot p) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.
- Decryption: *To decrypt a ciphertext $(\mathbf{u}, c)$ given the secret key $\mathbf{s}$, output the $z \in \mathbb{Z}_p$ such that $z \cdot p$ is closest to $c - \langle \mathbf{u}, \mathbf{s}\rangle$ modulo q.*

The main constraints on the parameters are given by the correctness requirement (α cannot be too large) and the hardness requirement (α should be large enough to invoke the worst-case lattice connections of [46,43]). These constraints are satisfied if the following inequalities hold true:

$$\frac{n}{q} = \frac{n}{p^2} \leq \alpha \leq \frac{1}{p \cdot \sqrt{m} \cdot \omega(\log n)} \tag{1}$$

By routine calculations, it is possible to satisfy the above inequalities for $m = O(n \log n)$, $p = \tilde{O}(\sqrt{mn})$, and $\alpha = 1/\tilde{O}(m \cdot \sqrt{n})$. This yields an underlying worst-case approximation factor of $\tilde{O}(n/\alpha) = \tilde{O}(n^{2.5})$ for lattice problems such as GapSVP.

Theorem 2. *For parameters satisfying Equation (1), the above cryptosystem is KDM-secure with respect to the set of affine functions over $\mathbb{Z}_p$, assuming that $\mathsf{LWE}_{q,\chi}$ is hard.*

3.3 Proof of Security

Overview. The proof of Theorem 2 has the following structure. First we show completeness, including correct decryption of key-dependent messages. Next we prove KDM security in two main steps.

The first step is to show that the view of the adversary in the real attack game may be generated faithfully, up to negligible statistical distance, via an alternate game: starting from the distribution $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$), the game invokes the transformation from Lemma 2 several times to produce independent distributions $A_{\mathbf{s}_1,\chi}, A_{\mathbf{s}_2,\chi}, \ldots$ for each user (where each $\mathbf{s}_i \leftarrow \chi^n$), and generates the users' public keys from these distributions in the natural way.

The transformation additionally outputs an invertible linear relation modulo q (hence modulo p as well) between each $\mathbf{s}_i$ and $\mathbf{s}$, thus linking every pair $\mathbf{s}_i, \mathbf{s}_j$ in a known way. The game answers the adversary's (key-dependent) message queries using these relations and the linear homomorphisms of the cryptosystem; this is where we use the fact that the system has a 'nice' ciphertext distribution. The crucial property of this game is that, aside from oracle access to $A_{\mathbf{s},\chi}$, the game works without needing to know any of the secret vectors $\mathbf{s}, \mathbf{s}_1, \mathbf{s}_2, \ldots$.

The second (and final) step is to consider a game that proceeds in exactly the same way as above, except that the original distribution $A_{\mathbf{s},\chi}$ is replaced by the *uniform* distribution $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. Because the game uses only oracle access to its given distribution, the two games are computationally indistinguishable under the assumption that $\mathsf{LWE}_{q,\chi}$ is hard (and by Lemma 1). Moreover, all the public keys in this game are uniform and independent, which implies that all the simulated ciphertexts are as well (up to negligible statistical distance). It follows that the adversary has negligible advantage in this game, and the scheme is KDM-secure.

Abstract Properties. Here we state a few technical facts about the cryptosystem. The proof of security relies only on these abstract properties, which can be shown via routine application of Gaussians over lattices from prior works (e.g., [40,46,25]). Due to space limitations, we defer the proofs to the full version.

The first fact is that the entries of the secret key may be represented unambiguously in the message space $\mathbb{Z}_p$. For convenience in dealing with key-dependent messages, from now on we view the secret key $\mathbf{s}$ as an element of $\mathbb{Z}_p^n \subset \mathbb{Z}_q^n$.

Lemma 3. *An $s \leftarrow \chi$ is of the form $s = (0, s_0) \in \mathbb{Z}_p \times \mathbb{Z}_p$ with overwhelming probability.*

Proof. This follows directly from the upper bound on α from Equation (1) and the exponential tail bound on the Gaussian distribution.

The following lemmas characterize the ciphertext distribution, which is needed for showing correctness, and (more importantly) for producing proper key-dependent ciphertexts using the scheme's homomorphisms.

Lemma 4. *With overwhelming probability over the choice of the public key $(\mathbf{A}, \mathbf{b})$ for secret key $\mathbf{s}$, the distribution $E_{\mathbf{A},\mathbf{b}}$ is within negligible statistical distance of $A_{\mathbf{s},\bar{\Psi}_\beta}$ for some $\beta \le \sqrt{2}r'$.*

Lemma 5. *Let $\mathbf{t} \in \mathbb{Z}_p^n$ and $y \in \mathbb{Z}_p$ be arbitrary. With overwhelming probability over the choice of the public key $(\mathbf{A}, \mathbf{b})$ for arbitrary secret key $\mathbf{s} \in \mathbb{Z}_p^n$, the following holds: for $(\mathbf{u}, v) \leftarrow E_{\mathbf{A},\mathbf{b}}$, the distribution of*

$$(\mathbf{u} - \mathbf{t} \cdot p, v + w \cdot p) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

is within negligible statistical distance of a (properly generated) encryption of the message $\langle \mathbf{t}, \mathbf{s} \rangle + w \in \mathbb{Z}_p$.

Finally, the next lemma is used for showing statistical security in the final hybrid game.

Lemma 6. *With overwhelming probability over the choice of a 'malformed' public key* $(\mathbf{A}, \mathbf{b})$ *from the* uniform *distribution* $U(\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m)$*, the distribution* $E_{\mathbf{A},\mathbf{b}}$ *is within negligible statistical distance of the uniform distribution* $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$*.*

Proof Details

Correctness. By Lemma 4, the noise in the c component of a ciphertext is distributed according to $\bar{\Psi}_\beta$ for some

$$\beta \le \sqrt{2}r' \le 4\alpha\sqrt{m} \cdot \omega(\sqrt{\log n}) \le \frac{1}{p \cdot \omega(\sqrt{\log n})},$$

by Equation (1). By the exponential tail inequality for Gaussians and the definition of $\bar{\Psi}_\beta$, the noise term does not exceed $q/2p = p/2$, except with negligible probability. We remark that the scheme can be made correct with probability 1 by modifying the key generation and encryption schemes to reject and re-sample values of $\mathbf{x}, \mathbf{r}, e$ that are 'too long;' however, this comes at the cost of an extra $\tilde{O}(\sqrt{n})$ factor in the noise parameter α and the underlying approximation factor for lattice problems.

The first hybrid game. We now describe an alternate game that faithfully simulates the true KDM attack game, up to negligible statistical distance. The game starts with access to the distribution $A_{\mathbf{s},\chi}$ for uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$. For each user i, it applies the transformation described in Lemma 2 (using fresh draws from $A_{\mathbf{s},\chi}$) to produce the distribution $A_{\mathbf{s}_i,\chi}$, where $\mathbf{s}_i$ is distributed according to χ^n. As a side-effect, the transformation also outputs invertible square matrices $\bar{\mathbf{A}}_i \in \mathbb{Z}_q^{n\times n}$ and vectors $\bar{\mathbf{b}}_i \in \mathbb{Z}_q^n$ such that for all i,

$$\mathbf{s} = \bar{\mathbf{A}}_i^{-T}(\bar{\mathbf{b}}_i - \mathbf{s}_i) \bmod q.$$

Note that by setting the right-hand sides equal for any i, j and reducing modulo p, we have

$$\bar{\mathbf{A}}_i^{-T}(\mathbf{s}_i - \bar{\mathbf{b}}_i) = \bar{\mathbf{A}}_j^{-T}(\mathbf{s}_j - \bar{\mathbf{b}}_j) \bmod p \iff \mathbf{s}_i = \bar{\mathbf{A}}_{i,j}^T \cdot \mathbf{s}_j + \bar{\mathbf{b}}_{i,j} \bmod p, \quad (2)$$

where $\bar{\mathbf{A}}_{i,j} = \bar{\mathbf{A}}_j^{-1}\bar{\mathbf{A}}_i$ and $\bar{\mathbf{b}}_{i,j} = \bar{\mathbf{b}}_i - \bar{\mathbf{A}}_{i,j}^T \cdot \bar{\mathbf{b}}_j$. The game then generates a public key $(\mathbf{A}_i, \mathbf{b}_i)$ for each user i in the usual way by drawing m samples from $A_{\mathbf{s}_i,\chi}$.

We now describe how the game answers (key-dependent) message queries. Suppose the adversary requests an encryption, under the jth user's public key $(\mathbf{A}_j, \mathbf{b}_j)$, of the function $f_{\mathbf{t},w}(\mathbf{s}_i) = \langle \mathbf{t}, \mathbf{s}_i \rangle + w \in \mathbb{Z}_p$ (for some $\mathbf{t} \in \mathbb{Z}_p^n, w \in \mathbb{Z}_p$) applied to the ith user's secret key $\mathbf{s}_i$. Observe that

$$f_{\mathbf{t},w}(\mathbf{s}_i) = \langle \mathbf{t}, \mathbf{s}_i \rangle + w = \underbrace{(\bar{\mathbf{A}}_{i,j} \cdot \mathbf{t})}_{\mathbf{t}' \in \mathbb{Z}_p^n}{}^T \cdot \mathbf{s}_i + \underbrace{\langle \mathbf{t}, \bar{\mathbf{b}}_{i,j} \rangle + w}_{w' \in \mathbb{Z}_p}.$$

The game therefore draws a sample $(\mathbf{u}, v) \leftarrow E_{\mathbf{A}_j,\mathbf{b}_j}$ and outputs

$$(\mathbf{u} - \mathbf{t}' \cdot p, v + w' \cdot p) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

This completes the description of the game.

By the above description and Lemmas 2 and 5, the following claim is apparent.

Claim. The views of the adversary in the real attack game and in the hybrid game are within negligible statistical distance.

The final hybrid game. The last hybrid game proceeds exactly as the one above, except that the initial distribution $A_{\mathbf{s},\chi}$ is replaced with $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. Note that the game above only treats $A_{\mathbf{s},\chi}$ as an oracle (it never uses $\mathbf{s}$ directly), so $A_{\mathbf{s},\chi}$ may be replaced in this way.

Now by Lemma 2, all the public keys $(\mathbf{A}_i, \mathbf{b}_i)$ generated by the game are uniform and independent. Moreover, by Lemma 6, all the (key-dependent) message queries are answered by ciphertexts that are uniform and independent of the message. The next claim follows, and the proof of Theorem 2 is complete.

Claim. Assuming that $\mathsf{LWE}_{q,\chi}$ is hard, the two hybrid games are computationally indistinguishable. Moreover, the adversary's advantage in the final hybrid game is negligible.

3.4 Amortized Extension

The system described in Section 3.2 encrypts only a single element $z \in \mathbb{Z}_p$ per syndrome $\mathbf{u} \in \mathbb{Z}_q^n$, so the ciphertext is a factor at least n larger than the message, and the encryption algorithm performs at least $n \cdot m$ operations per message element. Peikert, Vaikuntanathan, and Waters [44] proposed a significantly more efficient amortized version of the cryptosystem, which can encrypt $\ell = O(n)$ symbols using only about twice the time and space as the basic scheme. We can show that a variant of that system is also KDM-secure.

Construction 3. *Just as in Construction 1, the scheme is parametrized by $q = p^2$ and $\alpha \in (0, 1)$.*

– Key generation: *The secret key is $\mathbf{S} \leftarrow \chi^{n\times\ell}$. The public key is $(\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{m\times\ell}$ for $m \geq 2(n + \ell) \lg q$, where $\mathbf{B} = \mathbf{A}^T\mathbf{S} + \mathbf{X}$ for independent $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{X} \leftarrow \chi^{m\times\ell}$.*
– Encryption: *first define the distribution $E_{\mathbf{A},\mathbf{B}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$, obtained as follows: choose $\mathbf{r} \leftarrow D_{\mathbb{Z}^m,r}$ where $r = \omega(\sqrt{\log m})$, choose $\mathbf{e} \leftarrow \bar{\Psi}_{r'}^{\ell}$ where $r' = r \cdot \sqrt{\ell \cdot m} \cdot (\alpha + \frac{1}{2q})$, and output $(\mathbf{u}, \mathbf{v}) = (\mathbf{A}\mathbf{r}, \mathbf{B}^T\mathbf{r} + \mathbf{e})$.* Note that the parameter r' is a $\sqrt{\ell}$ factor larger than in Construction 1. *To encrypt a message $\mathbf{z} \in \mathbb{Z}_p^\ell$, draw $(\mathbf{u}, \mathbf{v}) \leftarrow E_{\mathbf{A},\mathbf{B}}$ and output $(\mathbf{u}, \mathbf{c} = \mathbf{v} + \mathbf{z} \cdot p)$.*
– Decryption: *output the $\mathbf{z} \in \mathbb{Z}_p^\ell$ such that $\mathbf{z} \cdot p$ is closest to $\mathbf{c} - \mathbf{S}^T\mathbf{u}$ modulo q.*

The proof of security extends to this construction in a straightforward way, with the exception of Lemma 4, which characterizes the ciphertext distribution and allows the simulator to answer key-dependent message queries faithfully. By generalizing the techniques from [46, Corollary 3.10] to higher dimensions, we can prove the following fact about $E_{\mathbf{A},\mathbf{B}}$, which suffices for proving KDM security.

Lemma 7. *With overwhelming probability over the choice of the public key* $(\mathbf{A}, \mathbf{B})$ *for secret key* $\mathbf{S}$*, the distribution* $E_{\mathbf{A},\mathbf{B}}$ *is within negligible statistical distance of* $(\mathbf{u}, \mathbf{S}^T\mathbf{u} + \mathbf{v})$*, where* $\mathbf{u} \in \mathbb{Z}_q^n$ *is uniform and* $\mathbf{v} \in \mathbb{Z}_q^\ell$ *is drawn from some distribution that depends only on* $\mathbf{B}$ *(and not on* $\mathbf{u}$*).*

Proof (Proof sketch). We need to show that the distribution of $\mathbf{X}^T\mathbf{r} + \mathbf{e} \in \mathbb{Z}_q^\ell$ conditioned on $\mathbf{A}\mathbf{r} = \mathbf{u}$ is essentially the same for every fixed $\mathbf{u}$. We can show that the distribution is a (discretized) *non-spherical* Gaussian whose covariance matrix depends only on r' and the positive semidefinite Gram matrix $\mathbf{X}^T\mathbf{X}$. The proof relies on the fact that a (continuous) Gaussian can be decomposed into the sum of two Gaussians whose covariance matrices sum to that of the original, and also uses the partial ordering of positive semidefinite matrices to establish a sufficient lower bound for r' (this is where the extra $\sqrt{\ell}$ term arises).

4 Linear-Stretch PRG in Quasi-Linear Time

4.1 Overview

Our starting point is a simple pseudorandom generator which was originally suggested in [10]. Let $G(\mathbf{A}, \mathbf{s}, r) = (\mathbf{A}, \mathbf{A}\mathbf{s} + e(r))$, where $\mathbf{A} \in \mathbb{Z}_2^{m \times n}, \mathbf{s} \in \mathbb{Z}_2^n$ and $e(\cdot)$ is a noise sampling procedure that uses a random input r to sample a random error vector from $\mathsf{Ber}_\varepsilon^m$. It was shown in [10] that, assuming the hardness of LPN_ε, the output distribution of G is pseudorandom. (See also [10,22,46,36,6]). In order to get expansion the noise-sampling algorithm should use a seed r of length shorter than m. Indeed, the noise vector can be sampled by using a seed r whose length is roughly $H_2(\varepsilon) \cdot m$, where H_2 is the binary entropy function. This gives an additive expansion of $m(1 - H_2(\varepsilon)) - n$ which is positive when the rate n/m is smaller than $1 - H_2(\varepsilon)$.

The resulting PRG is quite efficient as it mainly uses bit-operations rather than costly arithmetic operations over large fields. However, it still does not bring us to our goal (quasilinear time PRG). The main problem is that the matrix-vector product requires $\Omega(mn)$ operations, and so the time complexity of the generator is (at least) proportional to the product of the output length m and the security parameter n.

To solve this problem, we exploit the fact that the matrix $\mathbf{A}$ is public and hence can be reused with many different information words $\mathbf{s}_1, \ldots, \mathbf{s}_\ell$. Hence, the modified generator will compute the product of an $m \times n$ matrix $\mathbf{A}$ with an $n \times \ell$ matrix $\mathbf{S}$, and will add a noisy bit to each of the entries of the matrix $\mathbf{AS}$. By choosing ℓ carefully, we can use algorithms for fast rectangular matrix multiplication to speed up the computation.

We should also show how to sample the noise vector in quasilinear time *without using too many random bits.* At first glance, this seems to be hard, and indeed, we are not aware of any such sampling procedure[4]. However, we can bypass this problem by using a fast sampling procedure suggested in [5]. This procedure Sam samples an m-length noise vector e by using more than m random bits. To compensate this loss Sam also outputs a "leftover" vector – a vector v which is almost-random even when e is given. This allows us to concatenate v to the output of the PRG.

4.2 The Construction

The following lemma shows that for a random matrix $\mathbf{A}$, the mapping $(\mathbf{s}, e) \mapsto \mathbf{A}\mathbf{s} + e$ is pseudorandom even when it is applied to polynomially-many random strings $\mathbf{s}_1, \ldots, \mathbf{s}_\ell$. The proof combines the ideas of [10] with a standard hybrid argument and is therefore omitted from this version.

Lemma 8. *Let $0 < \varepsilon < \frac{1}{2}$ be a noise parameter and let $m(n), \ell(n)$ be arbitrary polynomials. If LPN_ε is hard, then the distribution $(\mathbf{A}, \mathbf{A}{\cdot}\mathbf{S}{+}\mathbf{E})$ is pseudorandom, where $\mathbf{A} \leftarrow \mathsf{U}_{m(n)\times n}$, $\mathbf{S} \leftarrow \mathsf{U}_{n\times\ell(n)}$, and $\mathbf{E} \leftarrow \mathsf{Ber}_\varepsilon^{m(n)\times\ell(n)}$.*

The following fact is based on [16].

Fact 4. *For every $r \le 0.172$ the product of a matrix in $\mathbb{Z}_2^{m\times m^r}$ and a matrix in $\mathbb{Z}_2^{m^r\times m}$ can be computed by a circuit of size $\tilde{O}(m^2)$.*

We will use a sampling procedure due to [5].

Lemma 9 (implicit in [5]). *There exist positive integers $k > 1$ and $c > 2k$, and a sampling algorithm* Sam *that uses $(k + k/c)N$ random bits and outputs a pair of strings $(\mathbf{e}, v)$ whose joint distribution is $2^{-\Omega(N)}$ statistically-close to $(\mathsf{Ber}_{2^{-k}}^N, \mathsf{U}_{kn})$. Moreover,* Sam *can be implemented in* $\mathbf{NC^0}$ *and therefore by a circuit family of size $O(N)$.*

We can now present our construction.

Construction 5. *Let $N = n^{12}$. Let k, c and* $\mathrm{Sam} : \{0,1\}^{(k+k/c)N} \to \{0,1\}^N \times \{0,1\}^{kN}$ *be the constants and sampling algorithm promised by Lemma 9. Let $e(r)$ and $v(r)$ denote the first and second entries of* $\mathrm{Sam}(r)$*. Define the function*

$$G(\mathbf{A}, \mathbf{S}, r) \overset{def}{=} (\mathbf{A}, \mathbf{A}\cdot\mathbf{S} + e(r), v(r))$$

where, $\mathbf{A} \in \mathbb{Z}_2^{n^6\times n}$, $\mathbf{S} \in \mathbb{Z}_2^{n\times n^6}$, $r \in \{0,1\}^{(k+k/c)N}$, $e(r)$ is parsed as a matrix in $\mathbb{Z}_2^{n^6\times n^6}$, and matrix addition is computed entry-wise.

Theorem 6. *Assuming that $\mathsf{LPN}_{2^{-k}}$ is hard, the function G defined in Construction 5 is a PRG with linear-stretch that can be computed by a circuit family of size quasilinear in the output length.*

[4] For example, the time complexity of the noise-sampling procedure of [22] is quadratic in the length of the error vector (for a constant error rate).

Proof. It can be easily verified that G takes less than $(k+0.6)N$ input bits and outputs more than $(k+1)N$ bits, and therefore, the stretch is linear in the input length. Pseudorandomness follows by Lemmas 9 and 8 as the tuple $(\mathbf{A}, \mathbf{A}\cdot\mathbf{S} + e(r), v(r))$ is statistically-indistinguishable from the tuple $(\mathbf{A}, \mathbf{A}\cdot\mathbf{S} + e(r), \mathsf{U}_{kn^7})$, which, in turn, is computationally-indistinguishable from $\mathsf{U}_{n^{4.5}+n^7+kn^7}$. Finally, by Fact 4, Lemma 9, and since entry-wise addition of two matrices is computable by linear-size circuits, the generator G can be computed by a circuit-family of size $\tilde{O}(N)$. $\square$

5 Weak Randomized PRF

An efficiently computable *randomized* function family $F : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}^{s(n)}$ is called a *randomized weak pseudorandom function* (RWPRF) if it satisfies the following:

– (weak pseudorandomness) For every polynomial $p(\cdot)$ the sequence

$$\big(A_1, F_S(A_1), \ldots, A_{p(n)}, F_S(A_{p(n)})\big) \text{ is pseudorandom,}$$

where $S \leftarrow \mathsf{U}_n$ and $(A_1, \ldots, A_{p(n)}) \leftarrow (\mathsf{U}_m)^{p(n)}$ and fresh internal randomness is used in each evaluation of F_S.

– (verifiability) There exists an efficient equality-tester algorithm V such that

$$\Pr[V(Y_1, Y_2) = \mathsf{equal}] > 1 - \mathrm{negl}(n)$$
$$\Pr[V(Y_1, Y_2') = \mathsf{not\text{-}equal}] > 1 - \mathrm{negl}(n),$$

where $S \leftarrow \mathsf{U}_n, A \leftarrow \mathsf{U}_m, A' \leftarrow \mathsf{U}_m, Y_1 \leftarrow F_S(A), Y_2 \leftarrow F_S(A)$, and $Y_2' \leftarrow F_S(A')$.

The PRG construction from the previous section, suggests a simple implementation of RWPRF. We let $\mathbf{S} \in \mathbb{Z}_2^{n\times \ell(n)}$ be the secret key of the function family, and let $\mathbf{A} \in \mathbb{Z}_2^{m(n)\times n}$ be the argument on which the function is being evaluated. The randomized function is defined as $f_{\mathbf{S}}(\mathbf{A}) = \mathbf{AS} + \mathbf{E}$ where $\mathbf{E} \in \mathsf{Ber}_\varepsilon^{m(n)\times \ell(n)}$ is a secret error vector which is randomly chosen in each invocation. By Lemma 8, the resulting function family is pseudorandom when it is evaluated on randomly chosen inputs $\mathbf{A}_1, \ldots, \mathbf{A}_q$. Also, given $y = f_{\mathbf{S}}(\mathbf{A})$ and $y = f_{\mathbf{S}}(\mathbf{B})$, one can easily check, with overwhelming probability, whether $\mathbf{A}$ and $\mathbf{B}$ are equal, even without knowing the key $\mathbf{S}$.

Note that now we have no limitation on the amount of randomness used to generate the error matrix $\mathbf{E}$. Hence, we can rely on the hardness of, say $\mathsf{LPN}_{1/4}$, and generate the error matrix $\mathbf{E}$ by taking the entry-wise product of 2 random matrices. The resulting function is quite efficient, and can be computed by a depth two Boolean circuit of size $O(n\ell m)$, or, by a circuit of size $\tilde{O}(m\ell)$ for a proper choice of the parameters. (The first option uses the trivial circuit for matrix multiplication, and the latter relies on Fact 4.)

When $\ell(n) = m(n)$ the function is symmetric, that is, one can replace the role of the argument and the key without violating the pseudorandomness property.

We also note that when $\ell(n)$ is sufficiently large (e.g., $\ell(n) > n/(1 - H_2(\varepsilon))$), then, except with negligible probability, $\mathbf{S}$ forms an error correcting code whose distance is larger than ε. In this case, the function $f_{\mathbf{S}}$ is injective and the equality-tester works well with respect to *every* input (as long as the collection-key and the internal randomness are random). By symmetry this is also true when the argument $\mathbf{A}$ is viewed as the key of the function. Hence, a random pair $(\mathbf{A}, f_{\mathbf{S}}(\mathbf{A}))$ forms a commitment to the collection key $\mathbf{S}$, which might be useful in some contexts.

Oblivious evaluation protocol. In an oblivious evaluation protocol for a collection of functions $f_{\mathbf{S}}$, one party (Alice) holds a key $\mathbf{S}$ and another party (Bob) holds a point $\mathbf{A}$. At the end of the protocol, Bob learns the value $f_{\mathbf{S}}(\mathbf{A})$, while Alice learns nothing. One can also consider the symmetric variant of the problem in which Alice learns $f_{\mathbf{S}}(\mathbf{A})$ and Bob learns nothing. In our setting, we also assume that the party who does not get the output selects the internal randomness of the function. That is, we consider the task of securely computing the following functionalities $g((\mathbf{S}, \mathbf{E}), \mathbf{A})) = (\lambda, \mathbf{AS} + \mathbf{E})$ and $h(\mathbf{S}, (\mathbf{A}, \mathbf{E})) = (\mathbf{AS} + \mathbf{E}, \lambda)$ where λ denotes the empty string. We give an efficient and secure protocol for evaluating both g and h Our protocol employs one-out-of-two oblivious transfer (OT) [20] for strings of length m. Such a protocol allows a receiver to receive one of two m-bit strings held by the sender in an oblivious way, that is, without revealing which string is selected.

Lemma 10. *There exists a constant-round protocol for securely evaluating f which uses circuits of size $O(m\ell n)$ with ℓn oracle gates to oblivious transfer which supports strings of length m.*

Proof. The protocol is similar to the protocol suggested in [23] for obliviously evaluating the Naor-Reingold PRF [42].

We begin with the version in which Alice receives the value of $f_{\mathbf{S}}(\mathbf{A})$. Let $\mathbf{S}$ be Alice's input and $\mathbf{A}, \mathbf{E}$ be Bob's input. For each $i \in [\ell]$ invoke in-parallel the following sub-protocol where $\mathbf{s}$ (resp. $\mathbf{e}$) is the i-th column of $\mathbf{S}$ (resp. $\mathbf{E}$):

– Bob chooses a random matrix $\mathbf{R} \leftarrow \mathbb{Z}_2^{m(n)\times n}$.
– For each $j \in [n]$ Alice and Bob call the string-OT oracle with Alice as the receiver and Bob as sender in the following way. Alice's input is $\mathbf{s}_j$, the j-th bit of $\mathbf{s}$, and Bob's input is the pair $(\mathbf{R}_j, \mathbf{R}_j + \mathbf{A}_j)$, where $\mathbf{R}_j$ and $\mathbf{A}_j$ are the j-th columns of $\mathbf{R}$ and $\mathbf{A}$. In addition, Bob sends the sum $\mathbf{t} = \mathbf{e} + \sum_j \mathbf{R}_j$.
– Alice sums up (over $\mathbb{Z}_2^{m(n)}$) the $n + 1$ vectors she received and outputs the result which is equal to $\sum_{\mathbf{s}_j=1} \mathbf{A}_j + \mathbf{e}$.

It is not hard to see that the protocol securely evaluates the functionality h. Indeed, the view of Alice which consists of the values learned by the OT and the vector $\mathbf{t}$ can be easily sampled given $f_{\mathbf{S}}(\mathbf{A}; \mathbf{E})$. A protocol in which Bob receives the output can be derived by slightly changing the previous protocol. Details omitted. □

Comparison to the OPRF of [23]. Let us briefly compare the efficiency of our scheme to the standard instantiation of OPRF [23] which is based on the Naor-Reingold function [42]. Our scheme uses large number of OT calls – if we set ℓ to be 1, which does not affect the security of the construction, this number is linear in the security parameter n. In contrast, the FIPR scheme uses only $O(m)$ calls where m is the length of the *input*. On the other hand, the additional overhead of FIPR is m modular multiplications and a single exponentiation, where our protocol performs only m vector additions ($O(mn)$ bit-wise XORs). This tradeoff is interesting as, by using the batch-OT protocol of [33], OT operations cost almost as little as symmetric operations. Furthermore, by using standard techniques one can compute all the OT operations in a preprocessing stage. In such case, it seems that the current scheme has the potential to obtain better performance, at least in some usage scenarios. (This possibility deserves further study.)

Application. Oblivious evaluation of pseudorandom function was recently used by Hazay and Lindell [30] to obtain an efficient two-party protocol for secure set-intersection (an explicit version for the semi-honest model appears in [23]). Our construction can be used in their protocol whenever the inputs of the parties are randomly distributed. This restriction is natural in some scenarios (e.g., when the inputs are names of entities or keys that were randomly selected by some authority) and can be always obtained at the expense of using a random oracle. We also note that RWPRF can be used to derive an identification scheme: we let parties share a key for the RWPRF and verify the identity of a party by querying the value of the function on a random point. When this protocol is instantiated with our function we get the well known HB protocol [32]. (This view is implicit in [36].)

6 Fast Circular-Secure Symmetric Encryption

6.1 The Construction

We now construct a symmetric encryption scheme. Our construction can be viewed as using the previous weak, randomized PRF in an analogous way to the standard construction of symmetric encryption from PRF, except that to deal with the error introduced by the PRF randomization we need to make the message redundant. This is done by employing an additional efficiently decodable error correcting code. As mentioned before, a similar construction was suggested in [26].

Let $\ell = \ell(n)$ be a message-length parameter which is set to be an arbitrary polynomial in the security parameter n. (Shorter messages are padded with zeroes.) Let $\varepsilon = 2^{-k}$ and $0 < \delta < 1$ be constants. We will use a family of good binary linear codes with information words of length $\ell(n)$ and block length $m = m(n)$, that has an efficient decoding algorithm D that can correct up to $(\varepsilon + \delta) \cdot m$ errors. We let $\mathbf{G} = \mathbf{G}_\ell$ be the $m \times \ell$ binary generator matrix of this family and we assume that it can be efficiently constructed (given 1^n).

Construction 7. *Let $N = N(n)$ be an arbitrary polynomial (which controls the tradeoff between the key-length and the time complexity of the scheme). The private key of the scheme is a matrix $\mathbf{S}$ which is chosen uniformly at random from $\mathbb{Z}_2^{n\times N}$.*

- Encryption*: To encrypt a message $\mathbf{M} \in \mathbb{Z}_2^{\ell\times N}$, choose a random $\mathbf{A} \leftarrow \mathbb{Z}_2^{m\times n}$ and a random noise matrix $\mathbf{E} \leftarrow \mathsf{Ber}_\varepsilon^{m\times N}$. Output the ciphertext*
$$(\mathbf{A}, \mathbf{A}\cdot\mathbf{S} + \mathbf{E} + \mathbf{G}\cdot\mathbf{M}).$$
- Decryption*: Given a ciphertext $(\mathbf{A}, \mathbf{Z})$ apply the decoding algorithm D to each of the columns of the matrix $\mathbf{Z} - \mathbf{AS}$ and output the result.*

Observe that the decryption algorithm errs only when there exists a column in $\mathbf{E}$ whose Hamming weight is larger than $(\varepsilon + \delta)m$, which, by Chernoff Bound, happens with negligible probability.

Quasilinear-time implementation. To get a quasilinear time implementation (for sufficiently long messages), we instantiate the above scheme with the error-correcting codes of Spielman [47, Thm. 19] which maps ℓ bits to $m = \Theta(\ell)$ bits with constant relative-distance and with the property that the encoding can be computed via a circuit of size $O(\ell)$ and the decoding can be decoded by a circuit of size $O(\ell \log \ell)$. Hence, the complexity of encryption (and decryption) is dominated by the complexity of the product $\mathbf{A}\cdot\mathbf{S}$. (The error matrix $\mathbf{E}$ can be generated in linear time by taking the entry-wise product of k random matrices $\mathbf{R}^{(1)}, \dots, \mathbf{R}^{(k)} \leftarrow \mathbb{Z}_2^{m\times N}$.) To compute this product in quasilinear time we set $N = n^6$ and assume that $m = \Omega(n^6)$, *i.e.,* assume that the message length $N\cdot\ell$ is at least $\Omega(n^{12})$. In this case, by Fact 4, the encryption and decryption can be computed by a circuit of size $\tilde{O}(N\ell)$.

Useful properties. The scheme enjoys several useful "homomorphic properties" which follow from its linear structure. In particular, given an encryption $(\mathbf{A}, \mathbf{Y})$ of an unknown message $\mathbf{M}$ under an unknown key $\mathbf{S}$, one can transform it to an encryption $(\mathbf{A}', \mathbf{Y}')$ of $\mathbf{M}+\mathbf{M}'$ under the key $\mathbf{S}+\mathbf{S}'$, for any given $\mathbf{M}', \mathbf{S}'$. This is done by letting $\mathbf{A}' = \mathbf{A}$ and $\mathbf{Y}' = \mathbf{Y} + \mathbf{AS}' + \mathbf{GM}'$. Furthermore, if the message $\mathbf{M}$ is the all zeroes string, then it is possible to convert the ciphertext $(\mathbf{A}, \mathbf{Y})$ to be an encryption $(\mathbf{A}', \mathbf{Y}')$ of the key $\mathbf{S}$ itself or, more generally, to be an encryption of $\mathbf{T}\cdot\mathbf{S}$ for an arbitrary linear transformation $\mathbf{T} \in \mathbb{Z}_2^{\ell\times n}$. This is done by letting $\mathbf{Y}' = \mathbf{Y}$ and $\mathbf{A}' = \mathbf{A} + \mathbf{G}\cdot\mathbf{T}$. Indeed, in this case $\mathbf{Y}' = \mathbf{A}'\mathbf{S} + \mathbf{E} + \mathbf{G}(\mathbf{TS})$. By choosing $\mathbf{T}$ to be the $\begin{pmatrix} \mathbf{I}_n \\ \mathbf{0}_{\ell-n\times n} \end{pmatrix}$, we can get an encryption of the key itself (padded with zeroes). We summarize these properties in the following lemma.

Lemma 11. *There exist efficiently computable transformations f, g, h such that for every* unknown $\mathbf{S} \in \mathbb{Z}_2^{n\times N}$ *and* $\mathbf{M} \in \mathbb{Z}_2^{\ell\times N}$ *and known* $\mathbf{S}' \in \mathbb{Z}_2^{n\times N}, \mathbf{M}' \in \mathbb{Z}_2^{\ell\times N}$ *and* $\mathbf{T} \in \mathbb{Z}_2^{\ell\times n}$*: $f(\mathbf{M}', \mathsf{Enc}_\mathbf{S}(\mathbf{M})) \equiv \mathsf{Enc}_\mathbf{S}(\mathbf{M} + \mathbf{M}')$, $g(\mathbf{S}', \mathsf{Enc}_\mathbf{S}(\mathbf{M})) \equiv \mathsf{Enc}_{\mathbf{S}+\mathbf{S}'}(\mathbf{M})$, and $h(\mathbf{T}, \mathsf{Enc}_\mathbf{S}(0^{\ell\times N})) \equiv \mathsf{Enc}_\mathbf{S}(\mathbf{TS})$, where $\mathsf{Enc}_\mathbf{K}(\mathbf{A})$ denotes a random encryption of the message $\mathbf{A}$ under the key $\mathbf{K}$.*

6.2 KDM Security

From now on, we fix the parameters $N(\cdot)$, $\ell(\cdot)$, $m(\cdot)$ and ε of our scheme. We consider the class of affine transformations that map the i-th column of the key $\mathbf{S}$ to the i-th column of the message $\mathbf{M}$. Let $t = t(n)$ be some arbitrary polynomial and let $N = N(n)$ and $\ell = \ell(n)$. For a matrix $\mathbf{T} \in \mathbb{Z}_2^{\ell \times n}$, a matrix $\mathbf{B} \in \mathbb{Z}_2^{\ell \times N}$ and an integer $i \in [t]$ we define the function $f_{T,B,i}$ which maps a tuple of t keys $(\mathbf{S}_1, \ldots, \mathbf{S}_t) \in (\mathbb{Z}_2^{n \times N})^t$ to a message $\mathbf{M} \in \mathbb{Z}_2^{\ell \times N}$ by letting $\mathbf{M} = \mathbf{T} \cdot \mathbf{S}_i + \mathbf{B}$. We let $\mathcal{F}_{\ell,N,t} = \{f_{\mathbf{T},\mathbf{B},i} | \mathbf{T} \in \mathbb{Z}_2^{\ell \times n}, \mathbf{B} \in \mathbb{Z}_2^{\ell \times N}, i \in [t]\}$. We will prove KDM-CPA-security with respect to the class $\mathcal{F}_{\ell,N,t}$. Formally,

Theorem 8. *Suppose that the* LPN_ε *is hard. Then Construction 7 is CPA-KDM secure with respect to* $\mathcal{F}_{\ell,N,t}$ *for every polynomial* $t(\cdot)$.

The proof uses the properties described in Lemma 11 in a straightforward way. A similar proof outline is used in [13].

Proof (Sketch). CPA security follows easily from Lemma 8. To prove KDM security, we show how to transform an adversary that wins the KDM game (with respect to $\mathcal{F}_{\ell,N,t}$) into an adversary that wins the standard CPA-game. Let $\mathbf{S}$ be the key of the scheme that was chosen by the challenger in the CPA game. The idea is to choose t random offsets $\mathbf{S}'_i \leftarrow U_{n \times N}$ and emulate the KDM game where the i-th key is $\mathbf{S}_i = \mathbf{S}'_i + \mathbf{S}$. Now, by using the properties of Lemma 11, we can transform a ciphertext $\mathsf{Enc}_{\mathbf{S}}(0^{\ell \times N})$ into a ciphertext $\mathsf{Enc}_{\mathbf{S}_j}(\mathbf{T} \cdot \mathbf{S}_i + \mathbf{B})$ for any given $i, j, \mathbf{T}$ and $\mathbf{B}$. Hence, we can perfectly emulate answers to the queries asked by the KDM adversary. □

As shown in [8], we can use the standard encrypt-then-MAC transformation to upgrade the security to KDM-CCA security (with respect to $\mathcal{F}_{\ell,N,t}$). In [34], it is shown that the existence of a linear-time computable MAC scheme follows from the existence of any one-way function. Hence, the intractability of LPN_ε allows us to construct a KDM-CCA-secure symmetric cryptosystem in which encryption and decryption are performed in quasilinear time, and the length of the ciphertext is linear in the message length.

Acknowledgement

The first author would like to thank Boaz Barak and Shai Halevi for helpful discussions, and Yuval Ishai for many valuable comments on an earlier draft of this paper.

References

1. Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness of formal encryption in the presence of key-cycles. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 374–396. Springer, Heidelberg (2005)

2. Albrecht, M., Bard, G., Hart, W.: Efficient multiplication of dense matrices over gf(2). CoRR, abs/0811.1714 (2008)
3. Alekhnovich, M.: More on average case vs approximation complexity. In: Proc. 44th FOCS, pp. 298–307 (2003)
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM J. Comput. 36(4), 845–888 (2006); Preliminary version in Proc. 45th FOCS (2004)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in NC^0. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 260–271. Springer, Heidelberg (2006)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography with constant input locality. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 92–110. Springer, Heidelberg (2007); full version in, `http://www.cs.princeton.edu/~bappelba/pubs/input-locality-full.pdf`
7. Backes, M., Dürmuth, M., Unruh, D.: OAEP is secure under key-dependent messages. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 506–523. Springer, Heidelberg (2008)
8. Backes, M., Pfitzmann, B., Scedrov, A.: Key-dependent message security under active attacks - BRSIM/UC-soundness of symbolic encryption with key cycles. In: CSF, pp. 112–124 (2007)
9. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
10. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994)
11. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)
12. Bogdanov, A., Mertens, M.C.: A parallel hardware architecture for fast gaussian elimination over gf(2). In: FCCM 2006: Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Washington, DC, USA, pp. 237–248. IEEE Computer Society, Los Alamitos (2006)
13. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008)
14. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. Cryptology ePrint Archive, Report 2008/375 (2008)
15. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
16. Coppersmith, D.: Rapid multiplication of rectangular matrices. SICOMP: SIAM Journal on Computing 11 (1982)
17. Damgård, I.B., Nielsen, J.B.: An efficient pseudo-random generator with applications to public-key encryption and constant-round multiparty computation (unpublished) (2002)
18. Dedic, N., Reyzin, L., Vadhan, S.P.: An improved pseudorandom generator based on hardness of factoring. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 88–101. Springer, Heidelberg (2003)
19. Dodis, Y., Kalai, Y.T., Lovett, S.: Cryptography with auxiliary inputs. In: Proc. 41st STOC (2009)

20. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. CACM: Communications of the ACM 28 (1985)
21. Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: New results for learning noisy parities and halfspaces. In: FOCS, pp. 563–574 (2006)
22. Fischer, J.-B., Stern, J.: An efficient pseudo-random generator provably as secure as syndrome decoding. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 245–255. Springer, Heidelberg (1996)
23. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005)
24. Gennaro, R.: An improved pseudo-random generator based on the discrete logarithm problem. J. Cryptology 18(2), 91–110 (2005)
25. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)
26. Gilbert, H., Robshaw, M.J.B., Seurin, Y.: How to encrypt with the LPN problem. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 679–690. Springer, Heidelberg (2008)
27. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. of the ACM 33, 792–807 (1986)
28. Haitner, I., Holenstein, T.: On the (im)possibility of key dependent encryption. In: TCC, pp. 202–219 (2009)
29. Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: CCS 2007, pp. 466–475 (2007)
30. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
31. Hofheinz, D., Unruh, D.: Towards key-dependent message security in the standard model. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 108–126. Springer, Heidelberg (2008)
32. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
33. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
34. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Proc. 40th STOC (2008)
35. Juels, A., Weis, S.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
36. Katz, J., Shin, J.S.: Parallel and concurrent security of the HB and HB^+ protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006)
37. Klivans, A.R., Sherstov, A.A.: Cryptographic hardness for learning intersections of halfspaces. In: FOCS, pp. 553–562 (2006)
38. Krause, M., Lucks, S.: On the minimal hardware complexity of pseudorandom function generators (extended abstract). In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 419–430. Springer, Heidelberg (2001)
39. Micciancio, D.: Improving lattice based cryptosystems using the Hermite normal form. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 126–145. Springer, Heidelberg (2001)

40. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007); Preliminary version in FOCS 2004 (2004)
41. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009)
42. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM 51(2), 231–262 (2004); Preliminary version in Proc. 38th FOCS (1997)
43. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC (2009)
44. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
45. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC, pp. 187–196 (2008)
46. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)
47. Spielman, D.A.: Linear-time encodable and decodable error-correcting codes. In: Proc. 27th STOC, pp. 388–397 (1995)

Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions

Brent Waters[⋆]

University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. We present a new methodology for proving security of encryption systems using what we call *Dual System Encryption.* Our techniques result in *fully* secure Identity-Based Encryption (IBE) and Hierarchical Identity-Based Encryption (HIBE) systems under the simple and established decisional Bilinear Diffie-Hellman and decisional Linear assumptions. Our IBE system has ciphertexts, private keys, and public parameters each consisting of a constant number of group elements. These results are the first HIBE system and the first IBE system with short parameters under simple assumptions.

In a Dual System Encryption system both ciphertexts and private keys can take on one of two indistinguishable forms. A private key or ciphertext will be *normal* if they are generated respectively from the system's key generation or encryption algorithm. These keys and ciphertexts will behave as one expects in an IBE system. In addition, we define *semi-functional* keys and ciphertexts. A semi-functional private key will be able to decrypt all normally generated ciphertexts; however, decryption will fail if one attempts to decrypt a semi-functional ciphertext with a semi-functional private key. Analogously, semi-functional ciphertexts will be decryptable only by normal private keys.

Dual System Encryption opens up a new way to prove security of IBE and related encryption systems. We define a sequence of games where we change first the challenge ciphertext and then the private keys one by one to be semi-functional. We finally end up in a game where the challenge ciphertext and all private keys are semi-functional at which point proving security is straightforward.

1 Introduction

The concept of Identity-Based Encryption (IBE) was first proposed by Shamir in 1984. In an IBE system a user can encrypt to another party simply by knowing

[⋆] Supported by NSF CNS-0716199, Air Force Office of Scientific Research (AFOSR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO) and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001.

their identity as well as a set of global parameters — eliminating the need to distribute a separate public key for each user in the system.

Although the concept received much interest, it wasn't until several years later that Boneh and Franklin [5] introduced the first Identity-Based Encryption scheme using groups with efficiently computable bilinear maps. The original Boneh and Franklin result used the random oracle heuristic to prove security under the Bilinear Diffie-Hellman assumption and a significant open question was whether the random oracle model could be removed.

Following the breakthrough result of Boneh and Franklin, there has been significant progress in realizing IBE in the standard model. First, Canetti, Halevi, and Katz [11] proved security without the random oracle heuristic, but under a weaker "Selective-ID" model where the attacker must declare the identity $\mathcal{I}^*$ that he will attack *before* even seeing the system's public parameters. Boneh and Boyen [2] then provided an efficient selectively secure scheme. Subsequently, Boneh and Boyen [3] and Waters [26] gave fully secure solutions in the standard model. The Waters scheme provided an efficient and provably fully secure system in the standard model under the decisional Bilinear Diffie-Hellman assumption; however, one drawback was that the public parameters consisted of $\mathcal{O}(\lambda)$ group elements for security parameter λ.

Partitioning Reductions. One very important common thread in all of the above systems is that they use what we call a *partitioning* strategy to prove security. In these systems, one proves security to an underlying complexity assumption by creating a reduction algorithm $\mathcal{B}$ that partitions the identity space into two parts — 1) identities for which it can create private keys; and 2) identities that it can use in the challenge ciphertext phase. This partitioning is embedded either in the public parameters at setup time in the standard model systems [11, 2, 3, 26] or programed into the random oracle [5]. In the selective model, systems the identity space can be "tightly" partitioned so that all the keys except $\mathcal{I}^*$ fall into the key creating partition, while reductions in fully secure systems will partition the space according to the number of private key queries $q(\lambda)$ that an attacker makes and the reduction "hopes" that the queries and challenge ciphertext identity fall favorably in the partition.

While the partitioning techniques have proved useful, they have two fundamental limitations. First, the most efficient fully secure and standard model IBE system due to Waters has large public parameters that might be impractical for some applications. The second and more compelling concern is that partitioning techniques appear to be inadequate for proving security of encryption systems that offer more functionality such as Hierarchical IBE [20, 18] and Attribute-Based Encryption [22] *even if we apply the random oracle model.* For instance, all known Hierarchical Identity-Based Encryption (HIBE) systems (in this vein) have an exponential degradation of security with the depth, n, of the hierarchy — rendering the security reductions meaningless for large n. The fundamental problem is that more advanced systems such as HIBE have more structure on the identity space that make (any known) partitioning strategies unusable. For example, in an HIBE system a partitioning reduction algorithm is constrained such that if it can create a private key for a particular identity vector then it must be able to for all of its descendants.

Moving beyond the partitioning paradigm. To overcome these obstacles, Gentry [15] proposed an IBE system with short public parameters that has a security reduction which moves beyond the partitioning paradigm. In his reduction the simulator is able to create a key for all identities and also use any identity as the challenge identity $\mathcal{I}^*$. At first glance, there is an apparent paradox in this strategy since it seems that the reduction algorithm could simply answer the challenge ciphertext itself by creating a private key for $\mathcal{I}^*$. To deal with this obstacle, Gentry's reduction algorithm can only generate *one* private key for each identity. For an attacker that makes at most q queries, the algorithm embeds a degree q polynomial $F(\cdot)$ and can create a private key with a tag component $F(\mathcal{I})$ for identity $\mathcal{I}$. The challenge ciphertext for $\mathcal{I}^*$ is structured such that it decrypts to the challenge message for the single key for $\mathcal{I}^*$ that the reduction could generate even though the message might be information theoretically hidden to an attacker with no knowledge of $F(\mathcal{I}^*)$.

Although the Gentry IBE achieved security in the standard model, it did so at the cost of using a significantly more complicated assumption called the decisional q-ABHDE assumption. In this assumption a generator g raised to several powers of an exponent a are given out (e.g., $g, g^a, g^{a^2}, \ldots, g^{a^q}$). In addition to the added complexity, the actual assumption used in the proof is *dependent* on the number of private key queries the adversary makes. This seems to be inherently tied to the need to embed the degree q polynomial f into a constant number group elements.

Interestingly, Gentry and Halevi [16] recently showed how to extend these concepts to get a fully secure HIBE system, although this system actually used an even more involved assumption. In addition, the "jump" from Gentry's IBE to the HIBE system added a significant amount of complexity to the system and proof of security.

Our Contribution. We present a new methodology for proving security of encryption systems using what we call *Dual System Encryption.* Our techniques result in *fully* secure IBE and HIBE systems under the simple and established decisional Bilinear Diffie-Hellman and decisional Linear assumptions. Our IBE system has ciphertexts, private keys, and public parameters each consisting of a constant number of group elements. Our results give the first HIBE system and the first IBE system with short parameters under simple assumptions.

Our conceptual approach departs significantly from both the partitioning paradigm and Gentry's approach. In a Dual System Encryption system, both ciphertexts and private keys can take on one of two indistinguishable forms. A private key or ciphertext will be *normal* if they are generated respectively from the system's key generation or encryption algorithm. These keys and ciphertexts will behave as one expects in an IBE system. In addition, we define *semi-functional* keys and ciphertexts. A semi-functional private key will be able to decrypt all normally generated ciphertexts; however, decryption will fail if one attempts to decrypt a semi-functional ciphertext with a semi-functional private key. Analogously, semi-functional ciphertexts will be decryptable only by normal private keys.

Dual System Encryption opens up a new way to prove security of IBE and related encryption systems. Intuitively, to prove security we define a sequence of games arguing that an attacker cannot distinguish one game from the next. The first game will be the real security game in which the challenge ciphertext and all private keys are distributed normally. Next, we switch our normal challenge ciphertext with a semi- functional one. We argue that no adversary can detect this (under our complexity assumption) since all private keys given can decrypt the challenge ciphertext regardless of whether it is normal or semi-functional. In the next series of games, we change the private keys one game at a time from normal to semi-functional, again arguing indistinguishability. In both the above proof arguments, our reduction algorithm $\mathcal{B}$ will be able to provide private keys for any identity and use any identity as a challenge identity — eliminating the need to worry about an abort condition. Finally, we end up in a game where the challenge ciphertext and all private keys are semi-functional. At this point proving security is straightforward since the reduction algorithm does not need to present any normal keys to the attacker and all semi-functional keys are useless for decrypting a semi-functional ciphertext.

The reader may have noticed one issue in our indistinguishability argument over private keys. If the reduction algorithm $\mathcal{B}$ wants to know whether a secret key $\mathrm{SK}_{\mathcal{I}}$ for $\mathcal{I}$ was semi-functional, couldn't it simply create a semi-functional ciphertext for $\mathcal{I}$ and test this itself (without using the attacker)? To deal with this issue our reduction algorithm embeds a degree one polynomial $F(\mathcal{I}) = A \cdot \mathcal{I} + B$ (over $\mathbb{Z}_p$). In each hybrid game the attacker can only create a semi-functional ciphertext for ciphertext identity $\mathcal{I}_c$ with a "tag" value of $\text{tag}_c = F(\mathcal{I}_c)$ and can only create a private key of unknown type for identity $\mathcal{I}_k$ with tag value of $\text{tag}_k = F(\mathcal{I}_k)$. Our system use the "two equation revocation" technique of Sahai and Waters [23] to enforce that the decryption algorithm will only work if the key tag and ciphertext tag are not equal. If the reduction algorithm attempted to test the key in question, decryption would fail unconditionally; and thus independently of whether it was a semi-functional key.[1].

In reflection, one reason our dual system achieves security from a simple assumption is that by changing the keys in small hybrid steps one by one we only need to worry about the relationship between the challenge ciphertext and one private key at a time. Our function F only needs to be able to embed a degree one polynomial; in contrast the Gentry reduction "takes on" all private keys at the same time and needs a complex assumption to embed a degree q polynomial.

HIBE and Other Encryption Systems. Building on our IBE system, we also provide a fully secure HIBE system. One remarkable feature is that the added complexity of the solution is rather small. Furthermore, our system combines

[1] Our core system has a negligible correctness error; however, we outline how to build a perfectly correct system in Section 4.

the structure of the Boneh-Boyen [2] selective-ID HIBE. This hints that we can leverage our methodology to adapt ideas from other selectively secure encryption systems (or those with complex assumptions) into fully secure ones under simple assumptions and also that prior selectively secure systems may have "lead us down the right path".

We believe that our Dual System methodology in the future will become a catalyst for proving adaptive security under simple assumptions for several other encryption systems including: Anonymous IBE and searchable encryption [4, 1, 10, 9, 24], Broadcast Encryption [14, 7], and Attribute-Based Encryption [22]. To add credence to this belief we give an adaptively secure broadcast system in the full version of our paper. Our broadcast system has ciphertext overhead of a constant number of group elements and is the first such system with a proof under a simple assumption.

Other Related Work. We note that there are remarkable IBE systems of Cocks [13] and Boneh, Gentry, and Hamburg [6] based on the quadratic residuosity assumption and Gentry, Peikert, and Vaikuntanathan [17] based on lattice assumptions. These systems are all proven secure under the random oracle heuristic.

Katz and Wang [21] gave an IBE system with a tight security reduction in the random oracle model using a two-key approach. One might view this as falling outside the partition approach, although their techniques do not appear to give a path to full security for HIBE and related problems.

2 Background

We present a few facts related to groups with efficiently computable bilinear maps and then define the decisional Billinear-Diffie-Hellman and decisional Linear Assumptions. For space considerations, the definitions of security for Identity-Based Encryption and Hierarchical Identity-Based Encryption are included in our full version.

2.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 Decisional Bilinear Diffie-Hellman Assumption

We define the decisional Bilinear Diffie-Hellman problem as follows. Choose a group $\mathbb{G}$ of prime order p, where the size of p is a function of the security parameters. Next, choose a random generator g and random exponents $c_1, c_2, c_3 \in \mathbb{Z}_p$. If an adversary is given

$$\boldsymbol{y} = g, g^{c_1}, g^{c_2}, g^{c_3},$$

it must remain hard to distinguish $e(g,g)^{c_1c_2c_3} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving decisional BDH problem in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\big(\boldsymbol{y}, T = e(g,g)^{c_1c_2c_3}\big) = 0\right] - \Pr\left[\mathcal{B}\big(\boldsymbol{y}, T = R\big) = 0\right] \right| \geq \epsilon \ .$$

Definition 1. *We say that the decisional BDH assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional BDH problem.*

2.3 Decisional Linear Assumption

We define the decisional Linear problem as follows. Choose a group $\mathbb{G}$ of prime order p, where the size of p is a function of the security paramters. Next, choose random generators g, f, ν and random exponents $c_1, c_2 \in \mathbb{Z}_p$. If an adversary is given

$$\boldsymbol{y} = g, f, \nu, g^{c_1}, f^{c_2},$$

it must remain hard to distinguish $\nu^{c_1+c_2} \in \mathbb{G}$ from a random element in $\mathbb{G}$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving decisional Linear problem in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\big(\boldsymbol{y}, T = \nu^{c_1+c_2}\big) = 0\right] - \Pr\left[\mathcal{B}\big(\boldsymbol{y}, T = R\big) = 0\right] \right| \geq \epsilon \ .$$

Definition 2. *We say that the decisional Linear assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional Linear problem.*

3 Identity-Based Encryption

We now present our core Identity-Based Encryption construction along with our proof of its security under the the decisional Linear and decisional BDH assumptions.

We first give the four algorithms of our IBE system. Next, we describe two additional algorithms for the creation of semi-functional ciphertexts and private keys respectively. The purpose of these algorithms is to define the structure of semi-functional ciphertexts and keys for our proof of security. We emphasize that these algorithms are not used in the actual system; indeed it is crucial for our security argument that no attacker could create ciphertexts or keys of this form.

Finally, we give the proof of our system against an attacker that makes at most q private key queries[2]. We organize our proof as a sequence of games. In the sequence, we will gradually change the actual security game; first by introducing a semi-functional challenge ciphertext and then introduce semi-functional private keys one by one. We show that under the decisional Linear Assumption no adversary can distinguish between each successive game. Finally, we end up in a game where the challenge ciphertext and the *all* the private keys given out are semi-functional. At this point we can prove security under decisional-BDH.

3.1 Construction

Setup(λ). The authority first chooses a group $\mathbb{G}$ of prime order p. Next, it chooses generators $g, v, v_1, v_2, w, u, h \in \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. Let $\tau_1 = vv_1^{a_1}, \tau_2 = vv_2^{a_2}$. It publishes the public parameters PK as the group description $\mathbb{G}$ along with:

$$g^b,\ g^{a_1},\ g^{a_2}, g^{b\cdot a_1},\ g^{b\cdot a_2},\ \tau_1, \tau_2, \tau_1^b, \tau_2^b,\ w,\ u,\ h, e(g,g)^{\alpha\cdot a_1\cdot b}.$$

The master secret key MSK consists of $g, g^{\alpha}, g^{\alpha\cdot a_1}, v, v_1, v_2$ as well as the public parameters. The identity space for the described scheme will be $\mathbb{Z}_p$, although we note in practice one can apply a collision resistant function to identities of arbitrary lengths.

*Encrypt(*PK*, $\mathcal{I}, M$).* The encryption algorithm chooses random s_1, s_2, t, and $\text{tag}_c \in \mathbb{Z}_p$. Let $s = s_1 + s_2$. It then blinds $M \in \mathbb{G}_T$ as $C_0 = M \cdot (e(g,g)^{\alpha a_1 \cdot b})^{s_2}$ and creates:

$$C_1 = (g^b)^{s_1+s_2},\ C_2 = (g^{b\cdot a_1})^{s_1},\quad C_3 = (g^{a_1})^{s_1},\ C_4 = (g^{b\cdot a_2})^{s_2},\ C_5 = (g^{a_2})^{s_2},$$

$$C_6 = \tau_1^{s_1}\tau_2^{s_2},\ C_7 = (\tau_1^b)^{s_1}(\tau_2^b)^{s_2} w^{-t}, E_1 = (u^{\mathcal{I}} w^{\text{tag}_c} h)^t,\ E_2 = g^t.$$

The ciphertext is $\text{CT} = C_0, \ldots, C_7, E_1, E_2, \text{tag}_c$.

*KeyGen(*MSK*, $\mathcal{I}$).* The authority chooses random $r_1, r_2, z_1, z_2, \text{tag}_k \in \mathbb{Z}_p$. Let $r = r_1 + r_2$.

Then it creates:

$$D_1 = g^{\alpha\cdot a_1} v^r.\quad D_2 = g^{-\alpha} v_1^r g^{z_1}.\quad D_3 = (g^b)^{-z_1}.\quad D_4 = v_2^r g^{z_2},\quad D_5 = (g^b)^{-z_2}$$

$$D_6 = g^{r_2\cdot b},\quad D_7 = g^{r_1},\quad K = (u^{\mathcal{I}} w^{\text{tag}_k} h)^{r_1}.$$

The secret key is $\text{SK} = D_1, \ldots, D_7, K, \text{tag}_k$.

[2] The maximum number of queries an attacker makes is, of course, a polynomial function $q(\cdot)$ of the security parameter; however, for notational simplicity we simply will speak of it making q private key queries.

Decrypt($\mathrm{CT}, K_{\mathcal{I}}$*).* The decryption algorithm will be able to decrypt a ciphertext encrypted for $\mathcal{I}$ with private key $\mathrm{SK}_{\mathcal{I}}$ if the ciphertext tag_c is not equal to the private key tag_k. Since both tags are chosen randomly, decryption will succeed with all but a negligible $1/p$ probability.

We break the decryption algorithm into a set of calculations. First, it computes:

$$\begin{aligned} A_1 &= e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4) \cdot e(C_5, D_5) \\ &= e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(v,g)^{b(s_1+s_2)r} e(v_1,g)^{a_1 b s_1 r} e(v_2,g)^{a_2 b s_2 r}. \end{aligned}$$

Recall that $r = r_1 + r_2$. Next, it computes

$$\begin{aligned} A_2 &= e(C_6, D_6) \cdot e(C_7, D_7) \\ &= e(v,g)^{b(s_1+s_2)r} e(v_1,g)^{a_1 b s_1 r} e(v_2,g)^{a_2 b s_2 r} \cdot e(g,w)^{-r_1 t}. \end{aligned}$$

Taking, $A_3 = A_1/A_2 = e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(g,w)^{r_1 \cdot t}$ leaves us with one more cancellation to get the message blinding factor. If $\mathrm{tag}_c \neq \mathrm{tag}_k$ then the decryption algorithm can compute

$$A_4 = \big(e(E_1, D_7)/e(E_2, K)\big)^{1/(\mathrm{tag}_c - \mathrm{tag}_k)} = e(g,w)^{r_1 \cdot t}.$$

Finally, we can recover the message by computing

$$C_0/(A_3/A_4) = M.$$

Altogether, decryption requires nine applications of the pairing algorithm.

3.2 Semi-Functional Algorithms

We now describe the semi-functional ciphertext and key generation algorithms. We will define them as algorithms that are executed with knowledge of the secret exponents; however, in a real system they will not be used. Their main purpose is to define the structures that will be used in our proof. We define both semi-functional ciphertexts and keys in terms of a transformation on a normal ciphertext or key.

Semi-Functional Ciphertexts. The algorithm first runs the encryption algorithm to generate a normal ciphertext CT for identity $\mathcal{I}$ and message M with $C'_1, \ldots, C'_7$,E'_1, E'_2. Then it chooses a random $x \in \mathbb{Z}_p$. It sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3, E_1 = E'_1$,$E_2 = E'_2$, leaving these elements and the tag_c unchanged. It then sets

$$C_4 = C'_4 \cdot g^{b a_2 x}, \quad C_5 = C'_5 \cdot g^{a_2 x}, \quad C_6 = C'_6 \cdot v_2^{a_2 x}, \quad C_7 = C'_7 \cdot v_2^{a_2 b x}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, E_2, \mathrm{tag}_c$.

Semi-Functional Secret Keys. The algorithm first runs the encryption algorithm to generate a normal private key $\mathrm{SK}_{\mathcal{I}}$ for identity $\mathcal{I}$ with $D'_1, \ldots, D'_7, K$. Then it chooses a random $\gamma \in \mathbb{Z}_p$. It sets $D_3 = D'_3, D_5 = D'_5, D_6 = D'_6, D_7 = D'_7, K = K'$, leaving these elements and the tag_k unchanged. It then sets

$$D_1 = D'_1 g^{-a_1 a_2 \gamma}, \quad D_2 = D'_2 \cdot g^{a_2 \gamma}, \quad D_4 = D'_4 \cdot g^{a_1 \gamma}$$

The semi-functional secret key is $\mathrm{SK} = D_1, \ldots, D_7, K, \mathrm{tag}_k$

Intuition. We make a few remarks about the nature of the semi-functional keys and the structure of the system. First, we note that if one attempted to decrypt a semi-functional ciphertext with a normal key, then the decryption would succeed. This follows from the fact that

$$e(g^{ba_2 x}, D_4) e(g^{a_2 x}, D_5) / \big(e(v_2^{a_2 x}, D_6) e(v_2^{a_2 bx}, D_7)\big) = 1$$

when D_4, D_5, D_6, D_7 come from a normally generated ciphertext. One can view this as the extra "random" space occupied by the semi-functional part of the ciphertext as being orthogonal to the space defined by a normal key. For similar reasons, the semi-functional components of a private key will not impede decryption when applied on a normal ciphertext. However, when a semi-functional key is used to decrypt a semi-functional ciphertext decryption will fail (or end up giving a random message) because an extra $e(g,g)^{-a_1 a_2 x \gamma b}$ will be multiplied by the intended message.

We note that in order to generate semi-functional ciphertexts and private keys (according to the defined procedures) one respectively needs $v_2^{a_2 b}$ and $g^{a_1 a_2}$ — neither of which is available from the public parameters.

3.3 Proof of Security

We organize our proof as a sequence of games. The first game defined will be the real identity-based encryption game and the last one will be one in which the adversary has no advantage unconditionally. We will show that each game is indistinguishable from the next (under a complexity assumption). As stated before, the crux of our strategy is to move to a security game where both the challenge ciphertext and private keys are semi-functional. At this point any keys the challenger gives out are not useful in decrypting the ciphertext. We first define the games as:

Game$_{\mathbf{Real}}$: The actual IBE security game defined in our full version.

Game$_i$: The real security game with the following two exceptions: 1) The challenge ciphertext will be a semi-functional ciphertext on the challenge identity $\mathcal{I}^*$. 2) The first i private key queries will return semi-functional private keys. The rest of the keys will be normal.

For an adversary that makes at most q queries we will be interested in **Game**$_0, \ldots,$ **Game**$_q$. We note that in **Game**$_0$ the challenge ciphertext is semi-functional, but all keys are normal and in **Game**$_q$ all private keys are semi-functional.

Game$_{\text{Final}}$: The real security game with the following exceptions: 1) The challenge ciphertext is a semi-functional encryption on a *random* group element of $\mathbb{G}_T$. 2) *All* of the private key queries result in semi-functional keys.

We now prove a set of Lemmas that argue about the distinguishablity of these games. For each proof we need to build a reduction simulator that both answers private key queries and creates a challenge ciphertext. We let **Game**$_{\text{Real}}\,\mathsf{Adv}_{\mathcal{A}}$ denote an algorithm $\mathcal{A}$'s advantage in the real game.

Lemma 1. *Suppose that there exists an algorithm $\mathcal{A}$ where* **Game**$_{Real}\,\mathsf{Adv}_{\mathcal{A}}$ $-$ **Game**$_0\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$*. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision Linear problem. We now describe how it executes the Setup, Key Phase, and Challenge phases of the IBE game with $\mathcal{A}$.

Setup. The algorithm chooses random exponents $b, \alpha, y_v, y_{v_1}, y_{v_2} \in \mathbb{Z}_p$ and random group elements $u, w, h \in \mathbb{G}$. It then sets $g = g, g^{a_1} = f, g^{a_2} = \nu$; intuitively a_1, a_2 are the exponents that the reduction cannot know itself.

Finally, it sets the variables as:

$$g^b,\ g^{b \cdot a_1} = f^b\ g^{b \cdot a_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}.$$

Using this it can calculate $\tau_1, \tau_2, \tau_1^b, \tau_2^b$ and $e(g,g)^{\alpha a_1 b} = e(g,f)^{\alpha \cdot b}$ in order to publish the public parameters PK. We also note that using α it can compute the master secret key for itself.

Key Generation Phases 1,2. Since $\mathcal{B}$ has the actual master secret key MSK it simply runs the key generation to generate the keys in both phases. Note that the MSK it has only allows for the creation of normal keys.

Challenge ciphertext. $\mathcal{B}$ receives two messages M_0, M_1 and challenge identity $\mathcal{I}^*$. It then flips a coin β. We describe the creation of the challenge ciphertext in two steps. First, it creates a normal ciphertext using the real algorithm by calling Encrypt(PK, $\mathcal{I}^*, M_\beta$), which outputs a ciphertext CT $= C_0', \ldots, C_7', E_1', E_2', \text{tag}_c$. Let s_1', s_2', t' be the random exponents used in creating the ciphertext.

Then we modify components of our ciphertext as follows. It sets

$$C_0 = C_0' \cdot \left(e(g^{c_1}, f) \cdot e(g, f^{c_2})\right)^{b \cdot \alpha}, \quad C_1 = C_1' \cdot (g^{c_1})^b, \quad C_2 = C_2' \cdot (f^{c_2})^{-b},$$

$$C_3 = C_3' \cdot (f^{c_2}), \quad C_4 = C_4' \cdot (T)^b, C_5 = C_5' \cdot T,$$

$$C_6 = C_6' \cdot (g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}, \quad C_7 = C_7' \cdot \left((g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}\right)^b,$$

$$E_1 = E_1', \quad E_2 = E_2'.$$

The returned ciphertext is CT $= C_0, \ldots, C_7, E_1, E_2, \text{tag}_c$.

If T is a tuple, then this assignment implicitly sets $s_1 = -c_2 + s'_1, s_2 = s'_2 + c_1 + c_2$, and $s = s_1 + s_2 = c_1 + s'_1 + s'_2$. If $T = \nu^{c_1+c_2}$ it will have the same distribution as a standard ciphertext; otherwise, it will be distributed identically to a semi-functional ciphertext. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

Lemma 2. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and $\mathbf{Game}_{\mathsf{k-1}}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{\mathsf{k}}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$ for some k where $1 \le k \le q$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision Linear problem. We now describe how it executes the Setup, Key Phase, and Challenge phases of the IBE game with $\mathcal{A}$.

Setup. Algorithm $\mathcal{B}$ first chooses random exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_u, y_h$. It then defines

$$g = g, g^b = f, g^{b\cdot a_1} = f^{a_1},\; g^{b\cdot a_2} = f^{a_2},\; v = \nu^{-a_1\cdot a_2},$$

$$v_1 = \nu^{a_2}\cdot g^{y_{v_1}},\; v_2 = \nu^{a_1}\cdot g^{y_{v_2}},\; e(g,g)^{\alpha\cdot a_1 b} = e(f,g)^{\alpha\cdot a_1}.$$

Now it can create

$$\tau_1 = vv_1^{a_1} = g^{y_{v_1}a_1} \quad \tau_2 = vv_1^{a_2} = g^{y_{v_2}a_2} \quad \tau_1^b = vv_1^{a_1} = f^{y_{v_1}a_1} \quad \tau_2^b = vv_1^{a_2} = f^{y_{v_2}a_2}.$$

Finally, $\mathcal{B}$ chooses random $A, B \in \mathbb{Z}_p$. It then sets

$$w = fg^{y_w}, \quad u = f^{-A}g^{y_u}, \quad h = f^{-B}g^{y_h}.$$

This will define all the public parameters of the system. Note that by virtue of knowing α, the algorithm $\mathcal{B}$ will know the regular master secret key.

We highlight the importance of the function $F(\mathcal{I}) = A\cdot\mathcal{I} + B$. One important feature is that for $\mathrm{tag}_c = F(\mathcal{I})$ we have $(u^{\mathcal{I}}w^{\mathrm{tag}_c}h) = f^{\mathrm{tag}_c - A\cdot\mathcal{I} - B}g^{\mathcal{I}\cdot y_u + y_h + \mathrm{tag}_c\cdot y_w} = g^{\mathcal{I}\cdot y_u + y_h + \mathrm{tag}_c\cdot y_w}$. In this case $\mathcal{B}$ will know the discrete log base g of the function. We also note that A, B are initially information theoretically hidden from the adversary. Since it is a pairwise independent function, if the adversary is given $F(\mathcal{I})$ for some identity, the, all values in $\mathbb{Z}_p$ are equally likely for $F(\mathcal{I}')$ for some $\mathcal{I} \neq \mathcal{I}'$.

Key Gen Phases 1,2. We break the Key Generation into three cases. Key Generation is done the same regardless of whether we are in phase 1 or 2.

Consider the i-th query made by $\mathcal{A}$.

Case 1: $i > k$
When i is greater than k our algorithm $\mathcal{B}$ will generate a normal key for the requested identity $\mathcal{I}$. Since it has the master secret key MSK it can run that algorithm.

Case 2: $i < k$
When i is less than k our algorithm $\mathcal{B}$ will generate a semi-functional key for the requested identity $\mathcal{I}$. It first creates a normal key using MSK. Then it makes it semi-functional using the procedure from above in Subsection 3.2. It can run this procedure since it knows $g^{a_1 a_2}$.

Case 3: $i = k$
The algorithm first runs the key generation algorithm to generate a normal private key $\mathrm{SK}_{\mathcal{I}}$ for identity $\mathcal{I}$ with $D'_1, \ldots, D'_7, K$ using $\mathrm{tag}_k{}^* = F(\mathcal{I})$. Let r'_1, r'_2, z'_1, z'_2 be the random exponents used.

It then sets

$$D_1 = D'_1 \cdot T^{-a_1 \cdot a_2},\ D_2 = D'_2 \cdot T^{a_2}(g^{c_1})^{y_{v_1}},\ D_3 = D'_3 \cdot (f^{c_2})^{y_{v_1}},\ D_4 = D'_4 \cdot T^{a_1}(g^{c_1})^{y_{v_2}},$$

$$D_5 = D'_5 \cdot (f^{c_2})^{y_{v_2}},\ D_6 = D'_6 \cdot f^{c_2},\ D_7 = D'_7 \cdot (g^{c_1}),\ K = K' \cdot (g^{c_1})^{\mathcal{I} \cdot y_u + y_h + \mathrm{tag}_k \cdot y_w}.$$

The semi-functional secret key is $\mathrm{SK} = D_1, \ldots, D_7, K, \mathrm{tag}_k$. We emphasize that the fact that $\mathrm{tag}_k = F(\mathcal{I})$ allowed us to created the component K. In addition, we note that we implicitly set $z_1 = z'_1 - y_{v_1} c_2$ and $z_2 = z'_2 - y_{v_2} c_2$ in order to be able to create D_2 and D_4.

If T is a Linear tuple of the form $T = \nu^{c_1 + c_2}$, then the k-th query results in a normal key under randomness $r_1 = r'_1 + c_1$ and $r_2 = r'_2 + c_2$. Otherwise, if T is a random group element, then we can write $T = \nu^{c_1 + c_2} g^{\gamma}$ for random $\gamma \in \mathbb{Z}_p$. This forms a semi-functional key where γ is the added randomness to make it semi-functional.

Challenge Ciphertext. Algorithm $\mathcal{B}$ is given a challenge identity $\mathcal{I}^*$ and messages M_0, M_1. Then it flips a coin β.

In this phase $\mathcal{B}$ needs to be able to generate a semi-functional challenge ciphertext. One problem is that $\mathcal{B}$ does not have the group element v_2^b so it cannot directly create such a ciphertext. However, in the case where $\mathrm{tag}_c{}^* = F(\mathcal{I}^*)$ it will have a different method of doing so.

$\mathcal{B}$ first runs the normal encryption algorithm to generate a normal ciphertext CT for identity $\mathcal{I}^*$ and message M^*; during this run it uses $\mathrm{tag}_c{}^* = F(\mathcal{I}^*)$. It then gets a standard ciphertext $C'_1, \ldots, C'_7, E'_1, E'_2$ under random exponents s'_1, s'_2, t'.

To make it semi-functional it chooses a random $x \in \mathbb{Z}_p$. It first sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$ leaving these elements and the $\mathrm{tag}_c{}^*$ unchanged. It then sets

$$C_4 = C'_4 \cdot f^{a_2 \cdot x}, \quad C_5 = C'_5 \cdot g^{a_2 \cdot x}, \quad C_6 = C'_6 \cdot v_2^{a_2 x}, \quad C_7 = C'_7 \cdot f^{y_{v_2} \cdot x \cdot a_2} \nu^{-a_1 \cdot x \cdot y_w \cdot a_2},$$

$$E_1 = E'_1 \cdot (\nu^{\mathcal{I} \cdot y_u + y_h + \mathrm{tag}_c \cdot y_w})^{a_1 a_2 x} \quad E_2 = E'_2 \cdot \nu^{a_1 a_2 \cdot x}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, E_2, \mathrm{tag}_c$.

Intuitively, the algorithm implicitly sets $g^t = g^{t'} + \nu^{a_1 a_2 x}$. This allows for the cancellation of the term $v_2^{a_1 a_2 b x}$ by w^{-t} in constructing C_7. Normally, this would be problematic for the generation of E_1; however since $\mathrm{tag}_c{}^* = F(\mathcal{I}^*)$ $\mathcal{B}$ is able to create this term.

If T is a tuple, then we are in $\mathbf{Game}_{k-1}$, otherwise we are in $\mathbf{Game}_k$. We highlight that the adversary cannot detect any special relationship between $\mathrm{tag}_c{}^*$ and $\mathrm{tag}_k{}^*$ since $F(\mathcal{I}) = A \cdot \mathcal{I} + B$ is a pairwise independent function and A, B are hidden from its view.

$\mathcal{B}$ receives a bit β' and outputs 0 if $\beta = \beta'$.

Lemma 3. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and $\mathbf{Game}_{\mathsf{q}}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{Final}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision BDH game.*

Proof. We give the proof of security in the full version of our paper.

Theorem 1. *If the decisional Linear and decisional BDH assumptions hold then no poly-time algorithm can break our IBE system.*

Proof. Any attacker's advantage in $\mathbf{Game}_{\mathrm{Final}}\,\mathsf{Adv}_{\mathcal{A}}$ in the final game must be 0 since it completely hides the bit β. By the sequence of games we established and Lemmas 1,2,3 an attacker's advantage in the real game $\mathbf{Game}_{\mathrm{Real}}\,\mathsf{Adv}_{\mathcal{A}}$ must be negligibly close to 0.

4 Discussion

In this section we discuss a few potential future variations and implications of our IBE system.

Achieving Perfect Correctness. Although having a negligible correctness error seems acceptable in practice, we would like to point out that we can close this gap by simply giving any user two private keys for an identity $\mathcal{I}$ each time they make a key request. The authority will simply run the original key generation algorithm twice with the restriction that the two key tags, $\mathrm{tag}_{kA}, \mathrm{tag}_{kB}$ are not equal. When attempting to decrypt a ciphertext at least one of the keys will work. The proof of security will work over each key piece — that is, each key request in the modified system will generate two distinct key requests (for the same identity) in the proof. We could also use a complementary two ciphertext approach and one private key approach.

Another potential solution is to run an efficient selectively secure IBE scheme [2] "in parallel". When a user encrypts a message M to $\mathcal{I}$ with tag_c in our original system, he will also encrypt M to the "identity" tag_c in the second selective system. A user with a key with tag_k will get a private key for "identity" tag_k in the second system. On decryption with $1 - 1/p$ probability the decryption algorithm will use the first ciphertext. However, if the tags align it can use the second ciphertext.

Signature Scheme. Naor[3] observed that any (fully secure) IBE system gives rise to a signature scheme secure under the same assumptions. The signature system from our IBE scheme has the favorable properties that the public parameters and signatures are a constant number of group elements, it is provable in the standard model, and it is stateless. While some previous signature schemes derived from IBE systems (e.g. BLS [8] or Waters [26] signatures) depended on the computational variants of the assumptions, our proof technique seems to require the decisional Linear Assumption. One interesting approach would be to see if one could create shorter signatures than those generated in the generic conversion by using the IBE systems private keys.

Chosen Ciphertext Security. We note that using the transformation of Canetti, Halevi, and Katz [12] we can achieve chosen ciphertext security from the HIBE scheme of Section 5.

Security under the XDH Assumption. One factor in the size and complexity of our IBE system is that it relies upon the Decisional Linear Assumption to hide the form of both keys and ciphertexts. One potential alternative is to use asymmetric bilinear groups, where we have $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Using these group we might assume DDH is hard both within $\mathbb{G}_1$ and within $\mathbb{G}_2$; this has also been called the XDH assumption. Using this assumption we might hope to shave off three group elements from both ciphertexts and private keys.

Alternative to Incompleteness. An critical part to arguing security is that an attacker could not distinguish normal keys from semi-functional ones. Our approach was to use a hybrid argument where for the key in question its $\text{tag}_k = F(\mathcal{I})$. If the simulator attempted to create the key in question for $\mathcal{I}^*$ and test it on the challenge ciphertext this would not work since $\text{tag}_k = \text{tag}_c$. Intuitively, the simulator could not test whether the key was semi-functional since decryption would fail *regardless* of whether the key was semi-functional or not. One might consider taking the opposite approach where decryption would always succeed if $\text{tag}_c = \text{tag}_k$ even if both the key and ciphertext are semi-functional. We note this approach would also require a slightly different proof strategy for proving Lemma 3.

5 Hierarchical Identity-Based Encryption

In this section we present our Hierarchical Identity-Based Encryption system. Our construction will build on top of our IBE scheme of Section 3. The reader will notice that the added complexity of moving from our IBE to HIBE system is remarkably simple. The same core concepts of our construction and proof methodology apply. One might view the HIBE system as "combining" the structure of the Boneh-Boyen [2] HIBE system with our techniques to get full security.

[3] The observation was documented by Boneh and Franklin [5].

One challenging aspect in the proof of security is that a private key of depth d will have associated tags: $\text{tag}_{k1}, \dots, \text{tag}_{kd}$. If we run our delegation algorithm to create a new key of depth $d+1$, the new key will inherit the previous key's tag values and there is no method for "re-randomizing" them. Most prior security definitions of HIBE [20, 18] define a game where all keys come from an authority and don't model any distinctions on how a key was created (i.e. trace paths of delegation). The prior definitions are only valid if keys from the delegation algorithm are distributed identically to a fresh call to the key generation algorithm [4]; however, due to the "tag lineage" described this is clearly not the case. To argue security we use a "complete" model of HIBE security introduced by Shi and Waters [25] that we define in our full version. Due to space considerations our proof of security is also in the full version.

5.1 Construction

In our system we will consider a hierarchical identity as an identity vector $\boldsymbol{I} = \mathcal{I}_1 : \cdots : \mathcal{I}_d$ for some depth d, where $d \le n$ for some maximum depth n. We assume that the identities are encoded such that for two identities $\boldsymbol{I}, \boldsymbol{I}'$ if $\mathcal{I}_i = \mathcal{I}'_i$ then $\mathcal{I}_j = \mathcal{I}'_j$ for all $j \le i$. We can enforce this by encoding all previous levels. For example, an identity of level one "com" and level two "yahoo" can be encoded as "com":"com.yahoo", where '.' is a special symbol. In practice, one will use a collision resistant hash function to hash identities of arbitrary length to $\mathbb{Z}_p$.

Setup(λ, n). The setup algorithm takes as input a security parameter and the maximum depth n. The authority first chooses a group $\mathbb{G}$ of prime order p. Next, it chooses generators $g, v, v_1, v_2, w, u_1, \dots, u_n, h_1, \dots, h_n \in \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$. It publishes the public parameters PK as the group description $\mathbb{G}$ along with:

$$g^b,\ g^{a_1},\ g^{a_2}, g^{b\cdot a_1},\ g^{b\cdot a_2},\ \tau_1, \tau_2, \tau_1^b, \tau_2^b, v,\ v_1,\ v_2,\ \ w,\ u_1, \dots, u_n,$$

$$h_1, \dots, h_n, e(g,g)^{\alpha\cdot a_1\cdot b}.$$

The master secret key MSK consists of $g, g^\alpha, g^{\alpha\cdot a_1}$ as well as the public parameters. The identity space for the described scheme will be $\mathbb{Z}_p$.

*Encrypt(*PK$, \boldsymbol{I} = \mathcal{I}_1 : \cdots : \mathcal{I}_d, M$*).* The encryption algorithm will encrypt to an identity vector of depth $d \le n$. It chooses random $s_1, s_2, t \in \mathbb{Z}_p$ and $\text{tag}_{c1}, \dots, \text{tag}_{cd} \in \mathbb{Z}_p$. Let $s = s_1 + s_2$. It then blinds $M \in \mathbb{G}_T$ as $C_0 = M \cdot (e(g,g)^{\alpha a_1 \cdot b})^{s_2}$ and creates:

$$C_1 = (g^b)^{s_1+s_2},\ C_2 = (g^{b\cdot a_1})^{s_1},\ C_3 = (g^{a_1})^{s_1},\ C_4 = (g^{b\cdot a_2})^{s_2},\ C_5 = (g^{a_2})^{s_2},$$

$$C_6 = \tau_1^{s_1}\tau_2^{s_2},\ C_7 = (\tau_1^b)^{s_1}(\tau_2^b)^{s_2} w^{-t},$$

$$E_1 = (u_1^{\mathcal{I}_1} w^{\text{tag}_{c1}} h_1)^t, \dots, E_d = (u_d^{\mathcal{I}_d} w^{\text{tag}_{cd}} h_d)^t, \quad \tilde{E} = g^t.$$

The ciphertext is $\text{CT} = C_0, \dots, C_7, E_1, , \dots, E_d, \tilde{E}, \text{tag}_{c1}, \dots, \text{tag}_{kd}$.

[4] This is actually the case for most prior systems, so the proofs of security do hold up.

*KeyGen(*MSK, $\boldsymbol{I} = \mathcal{I}_1 : \cdots : \mathcal{I}_d$*).* The authority chooses random $\mu_1, \ldots, \mu_d, r_2, z_1, z_2, \mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd} \in \mathbb{Z}_p$. First let $r_1 = \sum_{1 \le i \le d} \mu_i$ and then let $r = r_1 + r_2$. Then it creates:

$$D_1 = g^{\alpha \cdot a_1} v^r, \quad D_2 = g^{-\alpha} v_1^r g^{z_1}, \quad D_3 = (g^b)^{-z_1}, \quad D_4 = v_2^r g^{z_2}, \quad D_5 = (g^b)^{-z_2},$$

$$D_6 = g^{r_2 \cdot b}, \quad D_7 = g^{r_1},$$

$$(K_{1,1} = (u_1^{\mathcal{I}_1} w^{\mathrm{tag}_{k1}} h_1)^{\mu_1},\ K_{1,2} = g^{\mu_1}.), \ldots, (K_{d,1} = (u_d^{\mathcal{I}_d} w^{\mathrm{tag}_{kd}} h_d)^{\mu_d},\ K_{d,2} = g^{\mu_d})$$

The secret key is SK=$D_1, \ldots, D_7, (K_{1,1}, K_{1,2}), \ldots, (K_{d,1}, K_{d,2}) \mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd}$.

*Delegate(*PK, $\mathrm{SK}_{\boldsymbol{I} = \mathcal{I}_1 : \cdots : \mathcal{I}_d}, \mathcal{I}_{d+1}$*).* The algorithm will take a secret key SK $= D_1', \ldots, D_7'$, $(K_{1,1}', K_{1,2}')$, $\ldots, (K_{d,1}', K_{d,2}')$, $\mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd}$ for $\boldsymbol{I}$ and extend it to depth $d+1$ by creating a key for $\boldsymbol{I} : \mathcal{I}_{d+1}$.

The algorithm will "re-randomize" the existing key in the process of appending on a new key component; however, the existing tag_k values will remain. It chooses random $\mu_1, \ldots, \mu_{d+1}, r_2, z_1, z_2, \mathrm{tag}_{kd+1} \in \mathbb{Z}_p$. First let $r_1 = \sum_{1 \le i \le d+1} \mu_i$ and then let $r = r_1 + r_2$. Then it creates:

$$D_1 = D_1' \cdot v^r, \quad D_2 = D_2' \cdot v_1^r g^{z_1}, \quad D_3 = D_3' \cdot (g^b)^{-z_1}, \quad D_4 = D_4' \cdot v_2^r g^{z_2},$$

$$D_5 = D_5' \cdot (g^b)^{-z_2}, \quad D_6 = D_6' \cdot g^{r_2 \cdot b}, \quad D_7 = D_7' \cdot g^{r_1},$$

$$K_{1,1} = K_{1,1}' \cdot (u_1^{\mathcal{I}_1} w^{\mathrm{tag}_{k1}} h_1)^{\mu_1}, \ldots, K_{d,1} = K_{d,1}' \cdot (u_d^{\mathcal{I}_d} w^{\mathrm{tag}_{kd}} h_d)^{\mu_d},$$

$$K_{d+1,1} = (u_{d+1}^{\mathcal{I}_{d+1}} w^{\mathrm{tag}_{kd+1}} h_{d+1})^{\mu_{d+1}},$$

$$K_{1,2} = K_{1,2}' \cdot g^{\mu_1}, \ldots, K_{d,2} = K_{d,2}' \cdot g^{\mu_d}, \quad K_{d+1,2} = g^{\mu_{d+1}}.$$

The secret key is SK $= D_1, \ldots, D_7$,$(K_{1,1}, K_{1,2})$,. . .,$(K_{d+1,1}, K_{d+1,2})$,$\mathrm{tag}_{k1}, \ldots$,tag_{kd+1}.

*Decrypt(*CT, $K_{\mathcal{I}}$*).* The decryption algorithm will be able to decrypt a ciphertext encrypted for $\boldsymbol{I}'$ of depth d' with private key $\mathrm{SK}_{\boldsymbol{I}}$ of depth d if 1) $\forall i \le d : \ \boldsymbol{I}_i' = \boldsymbol{I}_i$ for all $i \le d$ and 2) $\forall i \le d : \ \mathrm{tag}_{ci} \ne \mathrm{tag}_{ki}$. We break the decryption algorithm into a set of calculations: First, it computes:

$$A_1 = e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4) \cdot e(C_5, D_5)$$

$$A_2 = e(C_6, D_6) \cdot e(C_7, D_7) \quad A_3 = A_1 / A_2 = e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(g,w)^{r_1 \cdot t}.$$

If $\forall i \le d$ we have $\mathrm{tag}_{ci} \ne \mathrm{tag}_{ki}$ then the decryption algorithm can compute

$$A_4 = \left(e(E_1, K_{1,2}) / e(\tilde{E}, K_{1,1})\right)^{1/(\mathrm{tag}_{c1} - \mathrm{tag}_{k1})} \cdots$$

$$\left(e(E_d, K_{d,2}) / e(\tilde{E}, K_{d,1})\right)^{1/(\mathrm{tag}_{cd} - \mathrm{tag}_{kd})} = e(g,w)^{t \sum_{1 \le d} \mu_i}.$$

Finally, we can recover the message by computing $C_0/(A_3/A_4) = M$.

References

[1] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)

[2] Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[3] Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)

[4] 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)

[5] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

[6] Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: FOCS, pp. 647–657 (2007)

[7] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)

[8] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)

[9] Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)

[10] Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)

[11] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)

[12] Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)

[13] Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA Int. Conf., pp. 360–363 (2001)

[14] Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)

[15] Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)

[16] Gentry, C., Halevi, S.: Hierarchical identity based encryption with polynomially many levels. In: TCC (2009)

[17] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)

[18] Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)

[19] Gentry, C., Waters, B.: Adaptive security in broadcast encryption sys- tems. In: Eurocrypt (2009)

[20] Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)

[21] Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM Conference on Computer and Communications Security, pp. 155–164 (2003)

[22] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

[23] Sahai, A., Waters, B.: Revocation systems with very small private keys. Cryptology ePrint Archive, Report 2008/309 (2008)

[24] Shi, E., Bethencourt, J., Chan, H.T.-H., Song, D.X., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symposium on Security and Privacy, pp. 350–364 (2007)

[25] Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)

[26] Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

The Group of Signed Quadratic Residues and Applications

Dennis Hofheinz[⋆] and Eike Kiltz[⋆⋆]

Abstract. We consider the cryptographic group of *Signed Quadratic Residues.* This group is particularly useful for cryptography since it is a "gap-group," in which the computational problem (i.e., computing square roots) is as hard as factoring, while the corresponding decisional problem (i.e., recognizing *signed* quadratic residues) is easy. We are able to show that under the factoring assumption, the Strong Diffie-Hellman assumption over the signed quadratic residues holds. That is, in this group the Diffie-Hellman problem is hard, even in the presence of a Decisional Diffie-Hellman oracle.

We demonstrate the usefulness of our results by applying them to the Hybrid ElGamal encryption scheme (aka Diffie-Hellman integrated encryption scheme — DHIES). Concretely, we consider the security of the scheme when instantiated over the group of signed quadratic residues. It is known that, in the random oracle model, the scheme is chosen-ciphertext (CCA) secure under the Strong Diffie-Hellman assumption and hence, by our results, under the standard factoring assumption. We show that furthermore, in the standard model, Hybrid ElGamal is CCA secure under the higher residuosity assumption, given that the used hash function is four-wise independent. The latter result is obtained using the recent "randomness extraction framework" for hash proof systems.

Keywords: Public-key encryption, chosen-ciphertext security, Hybrid ElGamal/DHIES.

1 Introduction

1.1 Quadratic Residues

The group of quadratic residues $\mathbb{QR}_N$ over a Blum integer $N = PQ$ (where $P \equiv Q \equiv 3 \bmod 4$) has proven to be a useful group for cryptographic purposes. For example, Rabin [30] proved that computing square roots in this group is equivalent to factoring the modulus N. The latter is believed to be hard in general ("factoring assumption"). Rabin's fundamental observation is the basis for a number of cryptographic protocols that are provably secure under the factoring assumption (e.g., the encryption and signature schemes [30,5,21]).

The quadratic residues have yet another useful property. Namely, given a uniformly random element modulo N (with Jacobi symbol 1), it is believed to be hard to decide whether the element is a square or not. This is the *quadratic*

[⋆] CWI, Amsterdam, `Dennis.Hofheinz@cwi.nl`. Work supported by NWO.
[⋆⋆] CWI, Amsterdam, `kiltz@cwi.nl`. Work supported by NWO.

residuosity assumption, a stronger assumption than the factoring assumption. On the bright side, there are again numerous cryptographic protocols whose security relies on the quadratic residuosity assumption (e.g., [18,11]).

However, the quadratic residuosity assumption also has a dark side. Namely, whenever an active adversary may choose group elements as protocol inputs (such as ciphertexts submitted for decryption), the receiving (honest) party may not be able to distinguish quadratic residues from quadratic non-residues. In particular, the adversary may learn some secret information by observing the protocol's different behaviour on quadratic residues and non-residues. Concretely, this problem naturally occurs when trying to reduce the chosen-ciphertext security (CCA security) of an encryption scheme (defined over the quadratic residues) to the factoring assumption. Specifically, during such a reduction, a decryption oracle has to be implemented without the knowledge of the factorization of N. Hence, the decryption oracle cannot distinguish quadratic residues from non-residues. This allows an adversary that uses the decryption oracle to submit, say, both $C \in \mathbb{Z}_N^*$ and $-C \in \mathbb{Z}_N^*$ (one of which is not a square) for decryption. This makes implementing a decryption oracle harder, in particular since the the submitted non-squares could be related to the challenge ciphertext.

Another intractability problem commonly used in cryptography is the Diffie-Hellman (DH) problem [13]. Given a generator g of a cyclic group $\mathbb{G}$ and $X = g^x, Y = g^y$, the DH key is defined as $\mathrm{DH}_g(X,Y) = g^{xy}$. The (Computational) DH problem is to compute $\mathrm{DH}_g(X,Y)$ from g, X, Y. For passive (chosen-plaintext) adversaries the security of the DH key exchange protocol [13] and the ElGamal encryption scheme [15] is equivalent to the DH problem. Over the group of quadratic residues (i.e., if $\mathbb{G} = \mathbb{QR}_N$), Shmuely [32] and McCurley [27] proved that the DH problem is at least as hard as factoring N.

The Strong Diffie-Hellman (SDH) problem [1] is to compute $\mathrm{DH}_g(X,Y)$ from g, X, Y while having access to a (Decisional) DH oracle that returns 1 on input $(\hat{Y}, \hat{Z})$ if $\mathrm{DH}_g(X, \hat{Y}) = \hat{Z}$ and $(\hat{Y}, \hat{Z}) \in \mathbb{G} \times \mathbb{G}$ (and 0 otherwise). Interestingly, for active (chosen-ciphertext) adversaries, the security of the (hashed) Diffie-Hellman key exchange protocol [13] and the Hybrid ElGamal encryption scheme [15] is equivalent to the SDH problem [9] in the random oracle model [3]. However, the result of Shmuely does not extend to prove that the SDH problem is at least as hard as factoring, since to simulate the DH oracle, one must be able to determine membership in the quadratic residues.

1.2 Signed Quadratic Residues

We propose to use a cryptographic group we call the *Signed Quadratic Residues* ($\mathbb{QR}_N^+$). This group has been suggested already by Fischlin and Schnorr in [16, Section 6] (in the different context of hard-core bits for generalized Rabin functions), but has not been investigated any further. This group is useful for cryptography since membership in $\mathbb{QR}_N^+$ can be publicly (and efficiently) verified while it inherits some nice intractability properties of the quadratic residues. For example, computing square roots in $\mathbb{QR}_N^+$ is also equivalent to factoring the modulus

N. We therefore have a "gap group" [29], in which the computational problem (i.e., computing a square root) is as hard as factoring, whereas the corresponding decisional problem (i.e., deciding if an element is a *signed* square) is easy. We can apply this observation to the Diffie-Hellman assumption. Namely, we extend Shmuely's result to show that in the group of signed quadratic residues, the *Strong* Diffie-Hellman problem is implied by the factoring assumption.

Concretely, the signed quadratic residues, $\mathbb{QR}_N^+$, are defined as the group $\mathbb{QR}_N^+ := \{|x| \,:\, x \in \mathbb{QR}_N\}$, where $|x|$ is the absolute value when representing elements of $\mathbb{Z}_N$ as the set $\{-(N-1)/2, \ldots, (N-1)/2\}$. We have that $(\mathbb{QR}_N^+, \circ)$ is a cyclic group, where the group operation is given by $a \circ b := |a \cdot b \bmod N|$. As already noted in [16], membership in $\mathbb{QR}_N^+$ can be efficiently verified since $\mathbb{QR}_N^+ = \mathbb{J}_N^+$, where $\mathbb{J}_N$ is the group of elements with Jacobi symbol 1 and $\mathbb{J}_N^+ := \{|x| \,:\, x \in \mathbb{J}_N\} = \mathbb{J}_N/\pm 1$.

1.3 Hybrid ElGamal over the Signed Quadratic Residues

The Hybrid ElGamal encryption scheme combines the original ElGamal encryption scheme with a hash function for key derivation and a symmetric cipher. As "Diffie-Hellman integrated encryption scheme" (DHIES) [1] it is contained in several standards bodies for public-key encryption, e.g., in IEEE P1363a, SECG, and ISO 18033-2. We consider the security of Hybrid ElGamal when implemented over the group of signed quadratic residues.

CCA SECURITY IN THE RANDOM ORACLE MODEL UNDER THE FACTORING ASSUMPTION. It is well known [1,12] that Hybrid ElGamal is CCA secure in the random oracle model under the SDH assumption. Recall that we show that the SDH assumption in the group of signed quadratic residues is implied by the factoring assumption. Hence, as an immediate application of our results, we obtain that Hybrid ElGamal over the signed quadratic residues is CCA secure in the random oracle model under the factoring assumption. (We emphasize that while the security proofs for Hybrid ElGamal from [1,12] are formulated for prime-order subgroups of $\mathbb{Z}_p^*$, they do *not* use knowledge about the order of the platform group, and hold literally in the group of signed quadratic residues.)

CCA SECURITY IN THE STANDARD MODEL UNDER THE HIGHER RESIDUOSITY ASSUMPTION. Using completely different techniques, we show the Hybrid ElGamal over the signed quadratic residues is CCA secure in the standard model under the *higher residuosity* assumption [19].[1] This result is obtained by applying the recent "randomness extraction framework" by [22] to a specific "high-entropic" hash proof system whose subset membership problem is hard assuming the higher residuosity assumption. We stress that this is the first security result for Hybrid ElGamal in the standard model from a non-interactive computational assumption.

[1] The higher residuosity assumption states that it is hard to distinguish random elements of $\mathbb{QR}_N$ from random elements of $\mathbb{G}_S$, where $\mathbb{G}_S$ is a subgroup of $\mathbb{QR}_N$ of unknown (large) order S.

1.4 Other Applications

Security of Diffie-Hellman key exchange. Similar to the Hybrid ElGamal scheme, the (hashed) Diffie-Hellman key exchange protocol [13] can be proven secure against active attacks in the random oracle model under the SDH assumption ([9, Theorem 5]). As with Hybrid ElGamal, the security proof does not use knowledge about the order of the platform group, and hence holds literally over the signed quadratic residues. In particular, we can employ our result about the SDH assumption in the group of signed quadratic residues. We get that the (hashed) Diffie-Hellman key exchange protocol is secure against active attacks in the random oracle model under the factoring assumption, when implemented over the signed quadratic residues.

Simplifying security proofs. As hinted above, encryption schemes that are already formulated over the quadratic residues have to take into account that the set of quadratic residues is not (or, rather, not known to be) efficiently recognizable. In particular, e.g., ciphertexts submitted for decryption may be non-squares. The usual way to deal with this problem is to first square the group elements supplied to decryption, and to "make up for this additional squaring" in the subsequent processing. Additionally, these works already propose to restrict the set of allowed ciphertexts to *signed* quadratic residues (e.g., to prevent an adversary to submit both C and $-C$ for decryption). Hence, the group of signed quadratic residues is implicitly used, but only to "transport" quadratic residues. Our proposal here is to work in the group of signed quadratic residues altogether, whenever a reduction to the factoring assumption is desired. Because the group of signed quadratic residues is efficiently recognizable, this avoids the extra squaring step and the connected complications. In particular, we can simplify both scheme and security proof of the CCA-secure encryption scheme from [21]. This results in a slight efficiency gain, since we save a few modular squarings. We stress that these modifications do not affect the actual reduction to factoring.[2]

1.5 Related Work

To the best of our knowledge, the group of signed quadratic residues appears first in [16] in the context of hard-core bits for generalized Rabin functions. Furthermore, as explained above, it has been used implicitly in several encryption schemes to "transport" quadratic residues, e.g., in [26,8,21]. The security of Hybrid ElGamal has been investigated in [12,25] in the random oracle model, and in [1] in the standard model. In particular, the latter work derives CCA security results for Hybrid ElGamal under the (interactive) "oracle Diffie-Hellman" assumption. The (non-interactive) computational assumption that we employ to show CCA security of Hybrid ElGamal has been suggested and used in [17,6],

[2] It is easy to see that squaring is a one-way permutation (as hard to invert as factoring N) also in the signed quadratic residues. Furthermore, the least significant bit of the squaring function (over the signed quadratic residues) is hard-core, see [16] who consider the "absolute Rabin function E_N^a."

also with the goal to construct hash proof systems for the use in encryption schemes. However, the encryption schemes from [17,6] are less efficient than Hybrid ElGamal due to the fact that they do not use randomness extraction techniques, but instead build on the Cramer-Shoup, resp. Kurosawa-Desmedt paradigms [11,23]. The paper [9] has a similar overall goal as ours. They propose the "Twin Diffie-Hellman" (2DH) assumption and show that the (interactive) *Strong* 2DH assumption is implied by the *standard* DH assumption. However, to be able to use this new assumption to prove security of the schemes of interest (among others also the Hybrid ElGamal and the Diffie-Hellman key-exchange protocol) they have to modify the schemes. Our results directly yield a security proof for the above schemes when instantiated in the specific group of signed quadratic residues.

2 Preliminaries

2.1 Notation

If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If $r \geq 1$ is a rational number then $[r] = \{1, \ldots, \lceil r \rceil\}$. If S is a set then $s \leftarrow_R S$ denotes the operation of picking an element s of S uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \leftarrow_R \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting z be the output. We write $\lg x$ for logarithms over the reals with base 2. The *min-entropy* of a random variable X is defined as $H_\infty(X) = -\lg(\max_{x \in \mathcal{X}} \Pr[X = x])$. If X is an element of a cyclic group $\mathbb{G} = \langle g \rangle$, we write $\mathrm{dlog}_g X$ for the smallest non-negative integer x with $X = g^x$.

2.2 Public-Key Encryption

A *public key encryption* scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}(k)$ consists of three polynomial time algorithms (PTAs), of which the first two, Kg and Enc, are probabilistic and the last one, Dec, is deterministic. Public/secret keys for security parameter $k \in \mathbb{N}$ are generated using $(pk, sk) \leftarrow_R \mathsf{Kg}(1^k)$. Given such a key pair, a message $m \in \mathcal{M}(k)$ is encrypted by $C \leftarrow_R \mathsf{Enc}(pk, m)$; a ciphertext is decrypted by $m \leftarrow_R \mathsf{Dec}(sk, C)$, where possibly Dec outputs $\perp$ to denote an invalid ciphertext. For consistency, we require that for all $k \in \mathbb{N}$, all messages $m \in \mathcal{M}(k)$, it must hold that $\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1$ where the probability is taken over the above randomized algorithms and $(pk, sk) \leftarrow_R \mathsf{Kg}(1^k)$.

The security we require for PKE is $\mathsf{IND\text{-}CCA}$ security [31,14]. We define the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as

$$\mathsf{Adv}^{\mathsf{cca}}_{\mathsf{PKE},\mathcal{A}}(k) \overset{\mathrm{def}}{=} \left| \Pr\left[\hat{b} = b \ : \begin{array}{l} (pk, sk) \leftarrow_R \mathsf{Kg}(1^k) \\ (m_0, m_1, St) \leftarrow_R \mathcal{A}_1^{\mathsf{Dec}(sk,\cdot)}(pk) \\ b \leftarrow_R \{0,1\} \, ; \, C^* \leftarrow_R \mathsf{Enc}(pk, m_b) \\ b' \leftarrow_R \mathcal{A}_2^{\mathsf{Dec}(sk,\cdot)}(C^*, St) \end{array} \right] - \frac{1}{2} \right| .$$

The adversary $\mathcal{A}_2$ is restricted not to query $\mathsf{Dec}(sk, \cdot)$ with C^*. PKE scheme PKE is said to be indistinguishable against chosen-ciphertext attacks ($\mathsf{IND\text{-}CCA}$ secure in short) if the advantage function $\mathsf{Adv}^{\mathrm{cca}}_{\mathsf{PKE},\mathcal{A}}(k)$ is a negligible function in k for all efficient $\mathcal{A}$.

2.3 Symmetric Encryption

A symmetric encryption scheme $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ is specified by its encryption algorithm E (encrypting $m \in \mathcal{M}(k)$ with keys $S \in \mathcal{K}_{\mathsf{SE}}(k)$) and decryption algorithm D (returning $m \in \mathcal{M}(k)$ or $\perp$). Here we restrict ourselves to deterministic algorithms E and D.

The most common notion of security for symmetric encryption is that of (one-time) ciphertext indistinguishability ($\mathsf{IND\text{-}OT}$), which requires that all efficient adversaries fail to distinguish between the encryptions of two messages of their choice. Another common security requirement is *ciphertext authenticity.* (One-time) ciphertext integrity ($\mathsf{INT\text{-}OT}$) requires that no efficient adversary can produce a new valid ciphertext under some key when given one encryption of a message of his choice under the same key. A symmetric encryption scheme which satisfies *both* requirements simultaneously is called secure in the sense of authenticated encryption ($\mathsf{AE\text{-}OT}$ secure). Symmetric ciphers secure in the sense of $\mathsf{AE\text{-}OT}$ can be constructed (following the encrypt-then-mac approach [2,12]) from a $\mathsf{IND\text{-}OT}$ secure symmetric encryption scheme and a MAC. Note that $\mathsf{AE\text{-}OT}$ security is a stronger notion than one-time chosen-ciphertext security ($\mathsf{IND\text{-}OTCCA}$) [2,12]. Formal definitions and constructions appear in, e.g., [20].

2.4 Hash Functions

Let $\mathcal{H}$ be a family of hash functions $\mathsf{H} : X \to Y$. With $|\mathcal{H}|$ we denote the number of functions in this family and when sampling from $\mathcal{H}$ we assume a uniform distribution. Let $k > 1$ be an integer, the hash-family $\mathcal{H}$ is k-wise independent if for any sequence of distinct elements $x_1, \ldots, x_k \in X$ the random variables $\mathsf{H}(x_1), \ldots, \mathsf{H}(x_k)$, where $\mathsf{H} \leftarrow_R \mathcal{H}$, are independent and uniformly random.

3 The Group of Signed Quadratic Residues

3.1 Quadratic Residues

An n-bit integer $N = PQ$ is called an RSA modulus if P and Q are two distinct $n/2$-bit odd primes. In what follows, we will assume that N is a Blum integer, i.e., an RSA modulus $N = PQ$ such that P and Q are both congruent 3 modulo 4. The group $\mathbb{Z}_N^*$ consists of all elements of $\mathbb{Z}_N$ that have an inverse modulo N. $\mathbb{Z}_N^*$ has order $\phi(N) = (P-1)(Q-1)$, where $\phi(N)$ is Euler's totient function. By $\mathbb{J}_N$ we denote the subgroup of all elements from $\mathbb{Z}_N^*$ with Jacobi symbol 1. $\mathbb{J}_N$ has index 2 in $\mathbb{Z}_N^*$ and has order $(P-1)(Q-1)/2$. Since N is Blum, $-1 \in \mathbb{J}_N$. By $\mathbb{QR}_N$ we denote the group of quadratic residues modulo N. Note that $\mathbb{QR}_N$ is a subgroup of $\mathbb{J}_N$ with index 2 and has order $(P-1)(Q-1)/4$. We remark that recognizing elements in $\mathbb{QR}_N$ is generally believed to be a hard problem (the quadratic residuosity problem).

3.2 Signed Quadratic Residues

Let N be an integer. For $x \in \mathbb{Z}_N$ we define $|x|$ as the absolute value of x, where x is represented as a signed integer in the set $\{-(N-1)/2, \ldots, (N-1)/2\}$. For a sub-group $\mathbb{G}$ of $\mathbb{Z}_N^*$ we define the "signed group", $\mathbb{G}^+$, as the group

$$\mathbb{G}^+ := \{|x| \ : \ x \in \mathbb{G}\}$$

with the following group operation. Namely, for $g, h \in \mathbb{G}^+$ and an integer x we define

$$g \circ h := |g \cdot h \bmod N|, \qquad g^{\underline{x}} := \underbrace{g \circ g \circ \ldots \circ g}_{x \text{ times}} = |g^x \bmod N| \, . \tag{1}$$

More complicated expressions in the exponents are computed modulo the group order, e.g., $g^{\underline{1/2}} = g^{\underline{2^{-1} \bmod \operatorname{ord}(\mathbb{G}^+)}}$. Note that taking the absolute value is a surjective homomorphism from $\mathbb{G}$ to $\mathbb{G}^+$ with trivial kernel if $-1 \notin \mathbb{G}$, and with kernel $\{-1, 1\}$ if $-1 \in \mathbb{G}$.

Let N be a Blum integer such that $-1 \notin \mathbb{QR}_N$. We will mainly be interested in $\mathbb{QR}_N^+$, which we call *signed quadratic residues* (modulo N). $\mathbb{QR}_N^+$ is a subgroup of $\mathbb{Z}_N^*/\pm 1$, with absolute values as a convenient computational representation. The following basic facts have already been noted in [16].

Lemma 1. *Let N be a Blum integer. Then:*
1. *$(\mathbb{QR}_N^+, \circ)$ is a group of order $\phi(N)/4$.*
2. *$\mathbb{QR}_N^+ = \mathbb{J}_N^+$. In particular, $\mathbb{QR}_N^+$ is efficiently recognizable (given only N).*
3. *If $\mathbb{QR}_N$ is cyclic, so is $\mathbb{QR}_N^+$.*

Proof. First, note that $|\cdot| : (\mathbb{Z}_N, \cdot) \to (\mathbb{Z}_N^+, \circ)$ is a group homomorphism so $(\mathbb{QR}_N^+, \circ)$ is a group. Since $-1 \notin \mathbb{QR}_N$, the map $\mathbb{QR}_N \to \mathbb{QR}_N^+$ has kernel $\{1\}$, and so $\operatorname{ord}(\mathbb{QR}_N^+) = \operatorname{ord}(\mathbb{QR}_N) = \phi(N)/4$. On the other hand, the map $\mathbb{J}_N \to \mathbb{J}_N^+$ has kernel $\{\pm 1\}$, and so $\operatorname{ord}(\mathbb{J}_N^+) = \operatorname{ord}(\mathbb{J}_N)/2 = \phi(N)/4$. Since $\mathbb{QR}_N \subseteq \mathbb{J}_N$, we have $\mathbb{QR}_N^+ \subseteq \mathbb{J}_N^+$, so $\operatorname{ord}(\mathbb{QR}_N^+) = \operatorname{ord}(\mathbb{J}_N^+)$ implies $\mathbb{QR}_N^+ = \mathbb{J}_N^+$. Elements in $\mathbb{QR}_N^+$ can be efficiently recognized since $\mathbb{QR}_N^+ = \mathbb{J}_N^+ = \mathbb{J}_N \cap [(N-1)/2]$. If $\mathbb{QR}_N$ is cyclic, a generator g of $\mathbb{QR}_N$ is mapped to a generator $|g|$ of $\mathbb{QR}_N^+$, so $\mathbb{QR}_N^+$ is a cyclic group.

3.3 Factoring Assumption

RSA INSTANCE GENERATOR. Let $0 \leq \delta < 1/2$ be a constant and $n(k)$ be a function. Let RSAgen be an algorithm that generates elements (N, P, Q), such that $N = PQ$ is an n-bit Blum integer and all prime factors of $\phi(N)/4$ are pairwise distinct and at least δn bit integers.[3]

[3] The "only large prime-factors" requirement is needed to ensure that the square of a random element in $\mathbb{Z}_N^*$ is a generator of $\mathbb{QR}_N$ with high probability $1 - O(2^{-\delta n(k)})$. The requirement that all prime factors are distinct ensures that $\mathbb{J}_N$ is cyclic.

Factoring assumption. The *factoring assumption* is that computing P, Q from N (generated by RSAgen) is hard. We write

$$\mathsf{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(k) := \Pr[\{P,Q\} \leftarrow_R \mathcal{A}(N) \ : \ (N,P,Q) \leftarrow_R \mathsf{RSAgen}(1^k)].$$

The factoring assumption for RSAgen holds if $\mathsf{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(k)$ is negligible for all efficient $\mathcal{A}$.

3.4 Strong Diffie-Hellman Assumption

Let $\mathbb{G}$ be a finite cyclic group whose order is not necessarily known. The Diffie-Hellman (DH) problem in $\mathbb{G}$ is to compute $\mathrm{DH}_g(X,Y) := g^{(\mathrm{dlog}_g X)(\mathrm{dlog}_g Y)}$ from $(\mathbb{G}, g, X, Y)$ for a uniform generator g and uniform $X, Y \in \mathbb{G}$. The strong Diffie-Hellman problem [1] is the same as the DH problem, but now the adversary has access to a Decision Diffie-Hellman oracle for fixed g and X, which is defined as $\mathrm{DDH}_{g,X}(\hat{Y},\hat{Z}) = 1$ if $\hat{Y}^{\mathrm{dlog}_g X} = \hat{Z}$ (and $\mathrm{DDH}_{g,X}(\hat{Y},\hat{Z}) = 0$ else), where $(\hat{Y},\hat{Z}) \in \mathbb{G} \times \mathbb{G}$. We do not define $\mathrm{DDH}_{g,X}$ in inputs $(\hat{Y},\hat{Z}) \notin \mathbb{G} \times \mathbb{G}$, since we assume that $\mathbb{G}$ is efficiently recognizable. For our purposes, we will consider the group $(\mathbb{QR}_N^+, \circ)$, i.e., the group of signed quadratic residues.

To an adversary $\mathcal{A}$ and RSAgen we associate

$$\mathsf{Adv}^{\mathrm{sdh}}_{\mathcal{A},\mathsf{RSAgen}}(k) := \Pr\left[Z = \mathrm{DH}_g(X,Y) \ : \ \begin{array}{l} (N,P,Q,S) \leftarrow_R \mathsf{RSAgen}(1^k)\,; \\ \text{unif. choose } g \text{ with } \langle g \rangle = \mathbb{QR}_N^+\,; \\ X, Y \leftarrow_R \mathbb{QR}_N^+\,; \\ Z \leftarrow_R \mathcal{A}^{\mathrm{DDH}_{g,X}(\cdot,\cdot)}(N,g,X,Y) \end{array} \right].$$

The Strong DH assumption holds relative to RSAgen if $\mathsf{Adv}^{\mathrm{sdh}}_{\mathcal{A},\mathsf{RSAgen}}(k)$ is negligible for all efficient $\mathcal{A}$.

Theorem 2. *If the factoring assumption holds then the strong DH assumption holds relative to* RSAgen*. In particular, for every strong DH adversary* $\mathcal{A}$*, there exists a factoring adversary* $\mathcal{B}$ *(with roughly the same complexity as* $\mathcal{A}$*) such that*

$$\mathsf{Adv}^{\mathrm{sdh}}_{\mathcal{A},\mathsf{RSAgen}}(k) \le \mathsf{Adv}^{\mathrm{fac}}_{\mathcal{B},\mathsf{RSAgen}}(k) + O(2^{-\delta n(k)}). \tag{2}$$

Proof. We construct $\mathcal{B}$ from a given $\mathcal{A}$. Concretely, $\mathcal{B}$ receives a challenge $N = PQ$, chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \setminus \mathbb{QR}_N^+$ and sets $h := u^{\underline{2}}$. Note that by definition of N, we have $\langle h \rangle = \mathbb{QR}_N^+$ except with probability $O(2^{-\delta n(k)})$. Then $\mathcal{B}$ chooses $a, b \in [N/4]$ and sets

$$g := h^{\underline{2}} \qquad\qquad X := h \circ g^{\underline{a}} \qquad\qquad Y := h \circ g^{\underline{b}}.$$

This implicitly defines

$$\mathrm{dlog}_g X = a + 1/2 \bmod \mathrm{ord}(\mathbb{QR}_N^+), \quad \text{and} \quad \mathrm{dlog}_g Y = b + 1/2 \bmod \mathrm{ord}(\mathbb{QR}_N^+),$$

where the discrete logarithms are of course considered in $(\mathbb{QR}_N^+, \circ)$. Again, by definition of N, the statistical distance between these (g, X, Y) and the input of

$\mathcal{A}$ in the strong DH experiment is bounded by $O(2^{-\delta n(k)})$. So $\mathcal{B}$ runs $\mathcal{A}$ on input (g, X, Y), and answers $\mathcal{A}$'s oracle queries $(\hat{Y}, \hat{Z})$ as follows. First, we may assume that $\hat{Y}, \hat{Z} \in \mathbb{QR}_N^+$ since (by Lemma 1) $\mathbb{QR}_N^+ = \mathbb{J}_N^+$ is efficiently recognizable. Next, since N is a Blum integer, the group order $\mathrm{ord}(\mathbb{QR}_N^+) = (P-1)(Q-1)/4$ is odd, and hence

$$\hat{Y}^{\underline{\mathrm{dlog}_g X}} = \hat{Z} \quad \Longleftrightarrow \quad \hat{Y}^{\underline{2\mathrm{dlog}_g X}} = \hat{Z}^{\underline{2}} \quad \Longleftrightarrow \quad \hat{Y}^{\underline{2a+1}} = \hat{Z}^{\underline{2}}.$$

Thus, $\mathcal{B}$ can implement the strong DH oracle by checking whether $\hat{Y}^{\underline{2a+1}} \overset{?}{=} \hat{Z}^{\underline{2}}$.

Consequently, with probability $\mathsf{Adv}^{\mathrm{sdh}}_{\mathcal{A},\mathsf{RSAgen}}(k) - O(2^{-\delta n(k)})$, $\mathcal{A}$ will finally output

$$Z = g^{\underline{(\mathrm{dlog}_g X)(\mathrm{dlog}_g Y)}} = g^{\underline{(a+1/2)(b+1/2)}} = h^{\underline{2ab+a+b+1/2}} \in \mathbb{QR}_N^+,$$

from which $\mathcal{B}$ can extract $v := h^{\underline{1/2}} \in \mathbb{QR}_N^+$ (using its knowledge about a and b). Since $u \notin \mathbb{QR}_N^+$ and $v \in \mathbb{QR}_N^+$ are two non-trivially different square roots of h, $\mathcal{B}$ can factor N by computing $\gcd(u - v, N)$.

4 Hybrid ElGamal over the Signed Quadratic Residues

We recall the Hybrid ElGamal (aka DHIES) scheme from [1,12]. There the scheme is described in a more general form over arbitrary cyclic groups. Here we restrict ourselves to the special case of $\mathbb{QR}_N^+$, for the following choice of N:

RSA Instance Generator. Let $0 \leq \delta \leq 1/4$ be a constant and $n(k)$ be a function. Let $\mathsf{RSAgen}' = \mathsf{RSAgen}'_{\delta,n(k)}$ be an algorithm that generates elements (N, P, Q, S), such that

- $N = PQ$ is an n-bit Blum integer such that the prime factors of $\phi(N)/4$ are pairwise distinct and at least δn-bit integers;
- $S > 1$ is a divisor of $\phi(N)/4$ with $1 < \gcd(S, (P-1)/2) < (P-1)/2$ and $1 < \gcd(S, (Q-1)/2) < (Q-1)/2$ (so S splits up into large prime factors of both $(P-1)/2$ and $(Q-1)/2$, but such that neither $(P-1)/2$ nor $(Q-1)/2$ divides S).

Note that by construction, $\gcd(S, \phi(N)/(4S)) = 1$. We stress that we need this choice of N *only* for the security proof of Hybrid ElGamal in the standard model. The security proof in the random oracle model (based on the hardness of factoring N) works with RSA instances as generated by RSAgen' or RSAgen.

4.1 The Encryption Scheme

Let $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a symmetric cipher with key-space $\{0,1\}^{\ell(k)}$, let $\mathcal{H} = (\mathcal{H}_k)_{k \in \mathbb{N}}$ be a family of hash functions with $\mathsf{H} : \{0,1\}^{2n(k)} \rightarrow \{0,1\}^{\ell(k)}$ for each $\mathsf{H} \in \mathcal{H}_k$. Define the following encryption scheme $\mathsf{DHIES} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$:

Key generation. $\mathsf{Kg}(1^k)$ chooses uniformly at random

- an RSA modulus $N = PQ$ generated with $\mathsf{RSAgen}'(1^k)$,
- a generator g of $\mathbb{QR}_N^+$,
- an exponent $x \in [N/4]$,
- a hash function $\mathsf{H} \in \mathcal{H}_k$.

Kg then sets $X = g^{\underline{x}} \in \mathbb{QR}_N^+$ and outputs a public key pk and a secret key sk, where

$$pk = (N, g, X, \mathsf{H}) \qquad\qquad sk = (N, x, \mathsf{H}).$$

Encryption. $\mathsf{Enc}(pk, m)$ chooses uniformly $y \in [N/4]$, sets

$$Y = g^{\underline{y}} \qquad\qquad K = \mathsf{H}(Y, X^{\underline{y}}) \qquad\qquad \psi = \mathsf{E}_K(m)$$

and outputs the ciphertext $(Y, \psi) \in \mathbb{QR}_N^+ \times \{0,1\}^*$.

Decryption. $\mathsf{Dec}(sk, (Y, \psi))$ verifies that $Y \in \mathbb{QR}_N^+$ and rejects if not. Then, Dec computes $K = \mathsf{H}(Y, Y^{\underline{x}})$ and outputs $\mathsf{D}_K(\psi)$.

Note that we present the DHIES scheme in a slightly generalized form for general symmetric ciphers SE, whereas in [1], SE consisted of a particular "encrypt-then-mac"-based cipher (which is AE-OT and therefore also IND-OTCCA secure).

4.2 Security

We now state our claims about the security of DHIES. We will prove that the same scheme DHIES is secure in the standard and in the random oracle model, under different assumptions.

Theorem 3. *Assume the factoring assumption holds for* $\mathsf{RSAgen}'_{n(k),\delta}$*,* $\mathcal{H}$ *is modeled as a random oracle, and* SE *is* IND-OTCCA *secure. Then* DHIES *is* IND-CCA *secure.*

[12, Theorem 9] show that the IND-CCA security of hashed ElGamal (viewed as a key encapsulation mechanism) in the random oracle model is implied by the strong DH assumption. In Theorem 9 (Appendix A) we formally show that their result does not use a specific group structure and can also be applied to our case. Putting Theorem 2 and Theorem 9 together yields Theorem 3. The following theorem will be proved in Section 5.

Theorem 4. *Assume the Higher Residuosity assumption (to be introduced in Section 5) holds relative to* $\mathsf{RSAgen}'_{\delta,n(k)}$*,* $\mathcal{H}$ *is a family of 4-wise independent hash functions, and* SE *is* AE-OT *secure with ℓ-bit keys. If $\delta n(k) \geq 4\ell$, then* DHIES *is* IND-CCA *secure.*

5 A Security Proof in the Standard Model

5.1 The Computational Hardness Assumption

To prove the security of DHIES in the standard model, we will make use of the following hardness assumption.

Let (N, P, Q, S) be generated by RSAgen'. We write $\mathbb{G}_S$ for the unique subgroup of order S of $\mathbb{Z}_N^*$. The higher residuosity (HR) assumption states that distinguishing a random element from $\mathbb{G}_S$ from a random element from $\mathbb{QR}_N$ is computationally infeasible. More formally, to an adversary and RSAgen' we associate

$$\mathsf{Adv}^{\mathrm{hr}}_{\mathcal{A},\mathsf{RSAgen}'}(k) := |\Pr[1 \leftarrow_R \mathcal{A}(N, g, c)] - \Pr[1 \leftarrow_R \mathcal{A}(N, g, \tilde{c})]| ,$$

where $(N, P, Q, S) \leftarrow_R \mathsf{RSAgen}'(1^k)$, $g, c \leftarrow_R \mathbb{G}_S$ and $\tilde{c} \leftarrow_R \mathbb{QR}_N$. The HR assumption for RSAgen' holds if $\mathsf{Adv}^{\mathrm{hr}}_{\mathcal{A},\mathsf{RSAgen}'}(k)$ is negligible for all efficient $\mathcal{A}$. Note that the HR assumption implicitly depends on the choice of $n(k)$ and δ. For concreteness, for $k = 80$ bits security one may choose $n(k) = 1024$ and $\delta = 1/8$. Then N can be sampled as $N = PQ$ for $P = 2P_SP_T + 1$ and $Q = 2Q_SQ_T + 1$ for primes P_S, P_T, Q_S, Q_T, with $P_S, Q_T \approx 2^{\delta n}$, such that for $S = P_SQ_S$, the order of $\mathbb{G}_S$ is about 2^{256}.

In the literature several related assumptions can be found. Closest to our assumption are the ones in [19,24,17,6] which are as our HR assumption but with a different distribution of N and/or using the groups $\mathbb{J}_N, \mathbb{Z}_N^*$ instead of $\mathbb{QR}_N$. Other similar assumptions were proposed in [18,10,24,4,28]. In all these assumptions the adversary is given (N, S) where $S \mid \phi(N)/4$, and has to distinguish a "random element" from one of the form $x^S \bmod N$.

5.2 A Variant of DHIES

To prove Theorem 4, we will consider a slightly different scheme, $\mathsf{DHIES}' = (\mathsf{Kg}', \mathsf{Enc}, \mathsf{Dec})$. It is defined as DHIES, with the only difference that in Kg', the element g from key generation is a uniform element from $\mathbb{G}_S^+$ (instead of an uniform element from $\mathbb{QR}_N^+$). The following lemma is immediate.

Lemma 5. *Under the HR assumption,* DHIES *is* $\mathsf{IND\text{-}CCA}$ *if and only if* DHIES' *is* $\mathsf{IND\text{-}CCA}$. *In particular, for every adversary $\mathcal{A}$ there exists an adversary $\mathcal{B}$ with*

$$|\mathsf{Adv}^{\mathrm{cca}}_{\mathsf{DHIES},\mathcal{A}}(k) - \mathsf{Adv}^{\mathrm{cca}}_{\mathsf{DHIES}',\mathcal{A}}(k)| \leq \mathsf{Adv}^{\mathrm{hr}}_{\mathsf{RSAgen}',\mathcal{B}}(k).$$

Lemma 6. *Under the conditions from Theorem 4,* DHIES' *is* $\mathsf{IND\text{-}CCA}$ *secure.*

A combination of the above two lemmas yields Theorem 4. The rest of this section is devoted to the proof of Lemma 6.

5.3 Hash Proof Systems

We recall the notion of hash proof systems introduced by Cramer and Shoup [11].

Smooth Projective Hashing. Let $\mathcal{C}, \mathcal{K}$ be sets and $\mathcal{V} \subset \mathcal{C}$ a language. In the context of public-key encryption (and viewing a hash proof system as a key-encapsulation mechanism (KEM) [12] with "special algebraic properties") one may think of $\mathcal{C}$ as the set of all *ciphertexts*, $\mathcal{V} \subset \mathcal{C}$ as the set of all *valid (consistent) ciphertexts*, and $\mathcal{K}$ as the set of all *symmetric keys*. Let $\Lambda_{sk} : \mathcal{C} \to \mathcal{K}$

be a hash function indexed with $sk \in \mathcal{SK}$, where $\mathcal{SK}$ is a set. A hash function Λ_{sk} is *projective* if there exists a projection $\mu : \mathcal{SK} \to \mathcal{PK}$ such that $\mu(sk) \in \mathcal{PK}$ defines the action of Λ_{sk} over the subset $\mathcal{V}$. That is, for every $C \in \mathcal{V}$, the value $K = \Lambda_{sk}(C)$ is uniquely determined by $\mu(sk)$ and C. In contrast, nothing is guaranteed for $C \in \mathcal{C} \setminus \mathcal{V}$, and it may not be possible to compute $\Lambda_{sk}(C)$ from $\mu(sk)$ and C. Following [22] we make the following two definitions about projective hash functions. The projective hash function is *κ-entropic* if for all $C \in \mathcal{C} \setminus \mathcal{V}$, $H_\infty(\Lambda_{sk}(C) \mid pk) \geq \kappa$ where in the above $pk = \mu(sk)$ for $sk \leftarrow_R \mathcal{SK}$. We furthermore define the collision probability as $\delta = \max_{C,C^* \in \mathcal{C}\setminus\mathcal{V}, C \neq C^*}(\Pr_{sk}[\Lambda_{sk}(C) = \Lambda_{sk}(C^*)])$.

Hash Proof System. A hash proof system $\mathsf{HPS} = (\mathsf{Par}, \mathsf{Pub}, \mathsf{Priv})$ consists of three algorithms. The randomized algorithm $\mathsf{Par}(1^k)$ generates parametrized instances of $par = (group, \mathcal{K}, \mathcal{C}, \mathcal{V}, \mathcal{PK}, \mathcal{SK}, \Lambda_{(\cdot)} : \mathcal{C} \to \mathcal{K}, \mu : \mathcal{SK} \to \mathcal{PK})$, where *group* may contain some additional structural parameters. The deterministic public evaluation algorithm Pub inputs the projection key $pk = \mu(sk)$, $C \in \mathcal{V}$ and a witness r of the fact that $C \in \mathcal{V}$ and returns $K = \Lambda_{sk}(C)$. The deterministic private evaluation algorithm Priv inputs $sk \in \mathcal{SK}$ and returns $\Lambda_{sk}(C)$, without knowing a witness. We further assume that μ is efficiently computable and that there are efficient algorithms given for sampling $sk \in \mathcal{SK}$, sampling $C \in \mathcal{V}$ uniformly (or negligibly close to) together with a witness r, sampling $C \in \mathcal{C}$ uniformly (given sk), and for checking membership in $\mathcal{C}$. Following [23] we also require that the subset membership problem can be efficiently solved with a master trapdoor.

Subset Membership Problem. As computational problem we require that the *subset membership problem* is hard in HPS. That is, for random $C_0 \in \mathcal{V}$ and random $C_1 \in \mathcal{C} \setminus \mathcal{V}$ the two elements C_0 and C_1 are computationally indistinguishable. This is captured by defining the advantage function $\mathsf{Adv}^{\mathrm{sm}}_{\mathsf{HPS},\mathcal{A}}(k)$ of an adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{sm}}_{\mathsf{HPS},\mathcal{A}}(k) \stackrel{\mathrm{def}}{=} \left|\Pr[1 \leftarrow_R \mathcal{A}(\mathcal{C}, \mathcal{V}, C_1)] - \Pr[1 \leftarrow_R \mathcal{A}(\mathcal{C}, \mathcal{V}, C_0)]\right|$$

where $\mathcal{C}$ is taken from the output of $\mathsf{Par}(1^k)$, $C_1 \leftarrow_R \mathcal{C}$ and $C_0 \leftarrow_R \mathcal{C} \setminus \mathcal{V}$.

5.4 IND-CCA Secure Encryption via Randomness Extraction

We recall the randomness extraction framework [22] that (building on [23,11]) transforms any κ-entropic HPS with hard subset membership problem into a IND-CCA secure encryption scheme.

Let $\mathsf{HPS} = (\mathsf{Par}, \mathsf{Pub}, \mathsf{Priv})$ be a hash proof system, let $\mathcal{H}$ be a family of hash functions with $\mathsf{H} : \mathcal{K} \to \{0,1\}^{\ell(k)}$ and let $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be an AE-OT secure symmetric encryption scheme with key-space $\mathcal{K}_{\mathsf{SE}} = \{0,1\}^{\ell(k)}$. We build a public-key encryption scheme $\mathsf{PKE}_{\mathsf{HPS}} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

Key generation. $\mathsf{Kg}(1^k)$ picks $par \leftarrow_R \mathsf{Par}(1^k)$, $sk \leftarrow_R \mathcal{SK}$ and defines $pk = \mu(sk) \in \mathcal{PK}$. Next, it picks a random hash function $\mathsf{H} \leftarrow_R \mathcal{H}$. The public-key is (par, H, pk), the secret-key is (par, H, sk).

Encryption. $\mathsf{Enc}(pk, m)$ picks $C \leftarrow_R \mathcal{V}$ together with its witness r that $C \in \mathcal{V}$. Session key $K = \mathsf{H}(\Lambda_{sk}(C)) \in \{0,1\}^{\ell}$ is computed as $K \leftarrow \mathsf{H}(\mathsf{Pub}(pk, C, r))$. The symmetric ciphertext is $\psi \leftarrow \mathsf{E}_K(m)$. The ciphertext is (C, ψ).

Decryption. $\mathsf{Dec}(sk, C)$ first checks if $C \in \mathcal{C}$ and rejects if not. Otherwise, it reconstructs the session key $K = \mathsf{H}(\Lambda_{sk}(C))$ as $K \leftarrow \mathsf{H}(\mathsf{Priv}(sk, C))$ and returns $\{m, \perp\} \leftarrow \mathsf{D}_K(\psi)$.

Theorem 7. *[22] Assume* HPS *is $\kappa(k)$-entropic with hard subset membership problem and negligible collision probability, $\mathcal{H}$ is a family of 4-wise independent hash functions with* $\mathsf{H} : \mathcal{K} \to \{0,1\}^{\ell(k)}$, *and* SE *is* $\mathsf{AE\text{-}OT}$ *secure. If $\kappa(k) \geq 2(\ell(k) + k)$ then* $\mathsf{PKE}_{\mathsf{HPS}}$ *is secure in the sense of* $\mathsf{IND\text{-}CCA}$.

5.5 A Hash Proof System for DHIES'

We now give a hash proof system HPS that yields the encryption scheme DHIES' via the transformation given in the last subsection. Define *group* $= (N, g)$, where $(N, P, Q, S) \leftarrow_R \mathsf{RSAgen}'(1^k)$ and g is a uniform generator of $\mathbb{G}_S^+$. Recall that N is of bit-length $n(k)$ and S is of bit-length $\delta n(k)$. Define $\mathcal{C} = \mathbb{QR}_N^+$ and $\mathcal{V} = \mathbb{G}_S^+ = \{g^{\underline{r}} : r \in \mathbb{Z}_S\}$. A value $r \in \mathbb{Z}$ is a witness of $C \in \mathcal{V}$. Note that it is possible to sample an almost uniform element from $\mathcal{V}$ together with a witness by first picking $r \in \mathbb{Z}_{[N/4]}$ and defining $C = g^{\underline{r}} \in \mathbb{G}_S^+$. Furthermore, membership in $\mathcal{C}$ can be efficiently checked by Lemma 1. Define $\mathcal{SK} = [N/4]$, $\mathcal{PK} = \mathbb{G}_S^+$, and $\mathcal{K} = \mathbb{QR}_N^+ \times \mathbb{QR}_N^+$ (which we interpret as a subset of $\{0,1\}^{2n(k)}$). For $sk = x \in [N/4]$, define $\mu(sk) = X = g^{\underline{x}} \in \mathbb{G}_S^+$. This defines the output of $\mathsf{Par}(1^k)$. For $C \in \mathcal{C}$ define

$$\Lambda_{sk}(C) := (C, C^{\underline{x}}) .$$

This defines $\mathsf{Priv}(sk, C)$. Given $pk = \mu(sk)$, $C \in \mathcal{V}$ and a witness $r \in \mathbb{Z}$ such that $C = g^{\underline{r}}$, public evaluation $\mathsf{Pub}(pk, C, r)$ computes $K = \Lambda_{sk}(C)$ as

$$K = (g^{\underline{r}}, X^{\underline{r}}) .$$

The trapdoor ω is the order of the group $\mathbb{G}_S$. This completes the description of HPS. Note that $\mathsf{PKE}_{\mathsf{HPS}}$ is exactly DHIES'. Therefore the proof Lemma 6 follows by combining Theorem 7 with the following.

Lemma 8. *Under the HR assumption, the subset membership problem is hard in* HPS. *Furthermore,* HPS *is $\delta n(k)$-entropic with collision probability $\delta = 0$.*

Proof. The subset membership problem is hard in HPS by definition of the HR assumption. The collision probability δ is zero since $\Lambda_{sk}(C) = (C, C^{\underline{x}})$ contains the element C. To show that HPS is $\delta n(k)$-entropic we consider an element $C \in \mathcal{C} \setminus \mathcal{V} = \mathbb{QR}_N^+ \setminus \mathbb{G}_S^+$. We can decompose $\mathbb{QR}_N^+$ as an internal direct product $\mathbb{QR}_N^+ = \mathbb{G}_T^+ \times \mathbb{G}_S^+$, where $\mathbb{G}_T^+$ is a cyclic group of order $T = (P-1)(Q-1)/(4S)$ with $\gcd(T, S) = 1$. Since T has only prime factors greater than $2^{\delta n(k)}$, and $C \notin$

$\mathbb{G}_S^+$, we have $\gcd(\mathrm{ord}(C), T) \geq 2^{\delta n(k)}$. Then, given N, g, $pk = \mu(sk) = X = g^{\underline{x}}$, and any $C \in \mathcal{C} \setminus \mathcal{V}$,

$$\begin{aligned} H_\infty((C, C^{\underline{x}}) \mid N, g, pk, C) &= H_\infty(C^{\underline{x}} \mid N, g, g^{\underline{x}}, C) \\ &= H_\infty(x \bmod \mathrm{ord}(C) \mid x \bmod S, S, T) \\ &\geq H_\infty(x \bmod \gcd(\mathrm{ord}(C), T) \mid x \bmod S, S, T) \\ &\overset{\gcd(S,T)=1}{=} H_\infty(x \bmod \gcd(\mathrm{ord}(C), T) \mid T) \geq \delta n(k) \ . \end{aligned}$$

This completes the proof.

5.6 Extensions

If one only requires a scheme that is IND-CCA secure in the standard model from the HR assumption, the one can turn encryption in DHIES′ slightly more efficient by choosing $y \leftarrow_R [2^{\delta n(k)+k}]$ (instead of $y \leftarrow_R [N/4]$). Furthermore, it is possible to prove the DHIES instantiated with RSAgen (instead of RSAgen′) IND-CCA secure under the ϕ-Hiding assumption [7] which essentially says that the two distributions (N, g) and (N', g') are computationally indistinguishable, where $(N, P, Q) \leftarrow_R$ RSAgen, $g \leftarrow_R \mathbb{QR}_N^+$ and $(N', P', Q', S') \leftarrow_R$ RSAgen′, $g' \leftarrow_R \mathbb{G}_{S'}^+$.

Acknowledgements. We thank Victor Shoup and the anonymous reviewers for useful comments.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993. ACM Press, New York (1993)
4. Benaloh, J.C.: Dense probabilistic encryption. In: SAC 1994, pp. 120–128 (1994)
5. Blum, M., Goldwasser, S.: An efficient probabilistic public-key encryption scheme which hides all partial information. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 289–299. Springer, Heidelberg (1985)
6. Brown, J., Nieto, J.M.G., Boyd, C.: Concrete chosen-ciphertext secure encryption from subgroup membership problems. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 1–18. Springer, Heidelberg (2006)
7. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)

8. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
9. Cash, D., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
10. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme (extended abstract). In: FOCS, pp. 372–382 (1985)
11. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
12. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
13. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
14. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
15. 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)
16. Fischlin, R., Schnorr, C.-P.: Stronger security proofs for RSA and Rabin bits. Journal of Cryptology 13(2), 221–244 (2000)
17. Gjøsteen, K.: Symmetric subgroup membership problems. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 104–119. Springer, Heidelberg (2005)
18. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
19. Groth, J.: Cryptography in subgroups of zn. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 50–65. Springer, Heidelberg (2005)
20. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
21. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, pp. 313–332. Springer, Heidelberg (2009)
22. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, pp. 589–608. Springer, Heidelberg (2009)
23. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
24. Kurosawa, K., Katayama, Y., Ogata, W., Tsujii, S.: General public key residue cryptosystems and mental poker protocols. In: Damgård, I. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 374–388. Springer, Heidelberg (1991)
25. Kurosawa, K., Matsuo, T.: How to remove MAC from DHIES. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 236–247. Springer, Heidelberg (2004)
26. Lucks, S.: A variant of the cramer-shoup cryptosystem for groups of unknown order. In: Zheng, Y. (ed.) ASIACRYPT 2002, vol. 2501, pp. 27–45. Springer, Heidelberg (2002)

27. McCurley, K.S.: A key distribution system equivalent to factoring. Journal of Cryptology 1(2), 95–105 (1988)
28. Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: ACM CCS 1998, pp. 59–66. ACM Press, New York (1998)
29. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
30. Rabin, M.O.: Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology (January 1979)
31. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
32. Shmuely, Z.: Composite diffie-hellman public-key generating systems are hard to break. Technical Report 356, Computer Science Department, Technion, Israel (1985)

A A Security Proof in the Random Oracle Model

Theorem 9. *([12,1]) If the strong DH assumption holds relative to* RSAgen', *and if* SE *is an* $\mathsf{IND\text{-}CCA}$ *secure symmetric cipher, then* DHIES *is* $\mathsf{IND\text{-}IND\text{-}OTCCA}$ *secure in the random oracle model. In particular, for every adversary* $\mathcal{A}$ *on* DHIES, *there exist adversaries* $\mathcal{B}$, *resp.* $\mathcal{B}'$ *on the strong DH assumption, resp. the* $\mathsf{IND\text{-}IND\text{-}OTCCA}$ *security of* SE, *such that* $\mathcal{B}$ *and* $\mathcal{B}'$ *have roughly the same complexity as* $\mathcal{A}$, *and*

$$\mathsf{Adv}^{\mathrm{cca}}_{\mathcal{A},\mathsf{DHIES}}(k) \leq \mathsf{Adv}^{\mathrm{sdh}}_{\mathcal{B},\mathsf{RSAgen}'}(k) + \mathsf{Adv}^{\mathrm{cca}}_{\mathcal{B}',\mathsf{SE}}(k) + O(2^{-\delta n(k)}).$$

The adaptations to [12, Theorem 9] are merely syntactic, and below we provide a short proof sketch. Putting Theorem 2 and Theorem 9 together yields Theorem 3.

Proof (Theorem 9). (Sketch.) We proceed in games.

Game 0. Let Game 0 be the original $\mathsf{IND\text{-}CCA}$ experiment with scheme DHIES and adversary $\mathcal{A}$. Here and in the following games, p_i denotes the probability that the experiment outputs 1, i.e., that $b = \hat{b}$, in Game i. By definition,

$$\mathsf{Adv}^{\mathrm{cca}}_{\mathcal{A},\mathsf{PKE}} = |p_0 - 1/2|. \tag{3}$$

Game 1. In Game 1, we modify the encryption of the challenge ciphertext (Y^*, ψ^*). Namely, now the symmetric ciphertext ψ^* is generated with an independent, uniform symmetric key K' as $\psi^* := \mathsf{E}_{K'}(m_b)$. Decryption queries of the form (Y^*, ψ) (for arbitrary $\psi \neq \psi^*$) are treated as if Y^* decrypted to key K' (and not key $K^* = \mathsf{H}(Y^*, Z^*)$ for $Z^* = Y^{*\underline{x}}$). Let F denote the event that $\mathcal{A}$ queries the random oracle H with (Y^*, Z^*). (F is defined in both Game 0 and Game 1.) Note that the views of $\mathcal{A}$ are *identical* in Game 0 and Game 1 unless F occurs. Hence,

$$|p_1 - p_0| \leq \Pr[F]. \tag{4}$$

Now we can build an adversary $\mathcal{B}$ on the strong DH assumption with

$$\Pr[F] \leq \mathsf{Adv}^{\mathrm{sdh}}_{\mathcal{B},\mathsf{RSAgen}'}(k) + O(2^{-\delta n(k)}). \tag{5}$$

Concretely, $\mathcal{B}^{\mathrm{DDH}_{g,X}(\cdot,\cdot)}(N, g, X, Y^*)$ simulates Game 1 with public key $pk := (N, g, X, \mathsf{H})$, and challenge ciphertext $(Y^*, \psi^*) := (Y, \mathsf{E}_{K'}(m_b))$. Adversary $\mathcal{A}$'s decryption queries $(\hat{Y}, \hat{\psi})$ are answered as follows (note that $\mathcal{B}$ does not know the secret key $x = \mathrm{dlog}_g X$, and hence cannot decrypt directly). If $\mathcal{A}$ has already made an H-query $\mathsf{H}(\hat{Y}, \hat{Z})$ for which $\mathrm{DDH}_{g,X}(\hat{Y}, \hat{Z}) = 1$, then $\hat{Z} = \hat{Y}^{\underline{x}}$, so the key $\hat{K} := \mathsf{H}(\hat{Y}, \hat{Z})$ can be used to decrypt $\hat{\psi}$. If on the other hand $\mathcal{A}$ made no such query, the hash value $\mathsf{H}(\hat{Y}, \hat{Z})$ for the "right" $\hat{Z} = \hat{Y}^{\underline{x}}$ has not yet been defined, and a symmetric key $\hat{K}$ can be freely invented (and then be used to decrypt $\hat{\psi}$). Note that in the latter case, care must be taken that once $\mathcal{A}$ makes an H-query $\mathsf{H}(\hat{Y}, \hat{Z})$ with $\mathrm{DDH}_{g,X}(\hat{Y}, \hat{Z}) = 1$ *later on*, then the right value $\hat{K}$ is returned.

If at any point, event F occurs, then $\mathcal{A}$ has submitted an H-query (Y, Z) for $Z = Y^x$ and effectively solved $\mathcal{B}$'s own DH challenge. This can be noticed by $\mathcal{B}$ (with the help of oracle $\mathrm{DDH}_{g,X}(\cdot,\cdot)$), and $\mathcal{B}$ can return Z. (5) follows. (A subtlety not yet mentioned is that X and Y^* are slightly differently distributed — but statistically close — in the strong DH experiment and in Game 0. This explains for the $O(2^{-\delta n(k)})$ term in (5).)

Game 2. We now change the symmetric part ψ^* of the challenge ciphertext into $\psi^* := \mathsf{E}_{K'}(R)$ for a uniform bit-string R of length $|m_0| = |m_1|$. Note that from Game 1 on, the symmetric key K' used to produce ψ^* is chosen independently. Furthermore, K' is only needed to perform decryptions of ciphertexts $\psi \neq \psi^*$ as required for ciphertexts (Y^*, ψ). Hence, we have

$$|p_2 - p_1| \leq \mathsf{Adv}^{\mathrm{cca}}_{\mathcal{B}',\mathsf{SE}}(k) \tag{6}$$

for a suitable $\mathsf{IND\text{-}CCA}$ adversary $\mathcal{B}'$ on SE.

On the other hand, $p_2 = 1/2$ since $\mathcal{A}$'s view in Game 2 is independent of b.

Putting (3,4,5,6) together yields the statement of Theorem 9.

Short and Stateless Signatures from the RSA Assumption

Susan Hohenberger[1,⋆] and Brent Waters[2,⋆⋆]

[1] Johns Hopkins University
susan@cs.jhu.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. We present the first signature scheme which is "short", stateless and secure under the RSA assumption in the standard model. Prior short, standard model signatures in the RSA setting required either a strong complexity assumption such as Strong RSA or (recently) that the signer maintain state. A signature in our scheme is comprised of one element in $\mathbb{Z}_N^*$ and one integer. The public key is also short, requiring only the modulus N, one element of $\mathbb{Z}_N^*$, one integer and one PRF seed.

To design our signature, we employ the known generic construction of fully-secure signatures from weakly-secure signatures and a chameleon hash. We then introduce a new proof technique for reasoning about weakly-secure signatures. This technique enables the simulator to predict a prefix of the message on which the adversary will forge and to use knowledge of this prefix to embed the challenge. This technique has wider applications beyond RSA.

We use it to provide an entirely new analysis of the security of the Waters signatures: the only short, stateless signatures known to be secure under the Computational Diffie-Hellman assumption in the standard model.

1 Introduction

Signature schemes are a fundamental building block of modern cryptography. As such, it is imperative to develop and to provide to practitioners efficient schemes in which we have the highest confidence of security. The focus of this work is, therefore, on designing "short" signatures secure under the weakest possible complexity assumptions in the standard model.

Most of today's short signature schemes can be divided into three categories: schemes that use random oracles (e.g., [10,26,22,2,23,4,15,14]), schemes that

⋆ Supported by NSF grant CNS-0716142 and a Microsoft New Faculty Fellowship.
⋆⋆ Supported by NSF CNS-0716199, Air Force Office of Scientific Research (AFOSR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO) and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. Portions of this work were done while this author was at SRI International.

require strong complexity assumptions (e.g., Strong RSA [13,7], q-Strong Diffie-Hellman [3] and LRSW [5]) and (recently) schemes where the signer must maintain state [18]. The one prior anomaly is the short and stateless signature scheme due to Waters [29], which is secure under the Computational Diffie-Hellman (CDH) assumption in bilinear groups in the standard model.

Our Contribution. We provide the first short and stateless signature scheme secure under RSA in the standard model. While there are several "standard" variants of the RSA assumption, the one we employ is that given a modulus N (chosen as the product of two large safe primes), a random exponent e less than and relatively prime to $\phi(N)$, and a random $y \in \mathbb{Z}_N^*$, it is hard to compute x such that $y = x^e \mod N$. (The restriction to safe primes can be removed.)

In our scheme, a signature is comprised of one element of $\mathbb{Z}_N^*$ and one integer. This is roughly the same size as the Strong RSA signatures of Gennaro, Halevi and Rabin [13]. The Strong RSA signatures of Cramer and Shoup [7] are even larger, requiring two elements in $\mathbb{Z}_N^*$ and one integer (for the basic scheme) or one element in $\mathbb{Z}_N^*$, one prime and one integer (for the trapdoor hash scheme). (We note that Fischlin [12] and Hofheinz-Kiltz [17] provide more efficient versions of Cramer-Shoup signatures.) Our public keys are also short, requiring only the modulus N, one element of $\mathbb{Z}_N^*$, one integer and one PRF seed. In contrast, the Waters' public keys are asymptotically larger, requiring $O(\lambda)$ group elements, where λ is the security parameter.

To realize our new construction, we introduce an entirely new proof technique for digital signatures, which we'll describe in detail shortly. We view this new technique as a major contribution of the work. To demonstrate its usefulness beyond RSA, we show that it can be applied in the CDH setting to obtain a variant of the Waters signatures [29].

Both of these signatures are also *online/offline* signatures, where the majority of the signer's computation can be performed offline before she knows the message. In Section 5, we discuss further computational optimizations and tradeoffs for our RSA scheme.

Intuition behind the Construction and Proof Technique. Our proof strategy begins with the previously-known method for constructing fully-secure signatures from weakly-secure signatures and a chameleon hash. Since chameleon hash functions exist under the hardness of factoring [19] and the RSA assumption [1,18], one only needs to design an appropriate weakly-secure scheme under RSA. Although, even this has proven an elusive task.

To design a weakly-secure scheme, we do as follows. Suppose the RSA challenge is (N, y, e^*) with the goal of computing $y^{1/e^*} \mod N$. Suppose the adversary provides us with the n-bit messages $M_1, \ldots, M_q$. Denote as w the shortest prefix of M^*, the message on which the adversary will later forge, that is different from all other prefixes of $M_1, \ldots, M_q$. Our strategy is to find w and then at this "point" embed the challenge exponent e^*. Of course, until the end of the game, the simulator does not know what M^* will be.

To find w, the simulator takes a guess as follows. If $q = 0$, meaning the adversary does not request any signatures, then the simulator only needs to guess the first bit of M^* and set w to this. If $q \geq 1$, the simulator may simply guess a pair (i^*, t^*), where $1 \leq i^* \leq q$ and $1 \leq t^* \leq n$. Interpret this pair as saying that M_{i^*} is a message with the longest prefix in common with M^* and the first location at which these two strings differ is t^*. (There may be more than one message in $M_1, \ldots, M_q$ containing the longest common prefix; guessing any one of them will suffice for our analysis.) If $q \geq 1$, then clearly, a valid pair (i^*, t^*) must exist and indeed, the simulator will have at least a $1/(qn)$ chance of guessing it.

Next we turn to embedding the challenge. We need to design a signature scheme that depends on all prefixes of its message. Let the public key contain the modulus N, a random $h \in \mathbb{Z}_N^*$ and a hash function H that maps arbitrary strings to prime numbers. Let $M^{(i)}$ denote the first i bits of M. For $i = 1$ to n, compute $e_i = H(M^{(i)})$. Then let the signature be

$$\sigma = h^{\prod_{i=1}^{n} e_i^{-1}} \mod N.$$

In the security proof, the simulator selects H so that $H(w) = e^*$. In other words, the simulator designs the public key so that the challenge exponent e^* is used in the forged signature on M^*, but in none of the signatures for $M_1, \ldots, M_q$. Thus, by properly setting h to be y raised to the product of all primes corresponding to all prefixes of $M_1, \ldots, M_q$, the simulator can answer its q signing queries and yet extract from the forgery the RSA solution $y^{1/e^*} \mod N$.

Brief Background on Short, Standard-Model Signatures. It is worthwhile to briefly compare our results to some short schemes in the standard model.

First, Dwork and Naor [9] and Cramer and Damgård [6] show how to make tree-based signatures shorter by using a "wide" tree (i.e., a larger branching factor) under the RSA assumption in the standard model. Roughly, there exists a trade-off where the tree depth can be decreased by a factor of $\lg w$ if the size of the public parameters is increased by a factor of w. However, the focus of this work is on finding even shorter signatures.

One approach has been to consider schemes under stronger complexity assumptions, such as Strong RSA [13,7], q-Strong Diffie-Hellman [3] and LRSW [5]. All of these schemes rely on the hardness of problems which, for any given instance, there are an *exponential* number of valid solutions. This stands in sharp contrast to problems such as RSA and CDH, where for any given instance, there is only *one* solution. Moreover, the latter two schemes require that the number of elements in the problem input grows with the number of signing queries made by the adversary. For instance, the q-Strong Diffie-Hellman assumption requires that given a generator g of prime order p and the tuple $(g^x, g^{x^2}, \ldots, g^{x^q})$, it is hard to compute $(c, g^{1/(x+c)})$ for any $c \in \mathbb{Z}_p^*$. Thus, if the adversary asks for q signatures, then the problem must remain hard when q powers of x are released. In RSA and CDH (and Strong RSA), the number of input elements is always a small constant, independent of the adversary's behavior. To obtain high confidence in the security of our schemes, we should based them on the simplest and weakest assumptions possible. In fairness, these excellent works have laid the foundation of our result and they are still unrivaled in their computational efficiency by our RSA scheme. Now that we have "short" RSA signatures, it would be of great interest to reduce the cost of signing and verification. See Section 5.

Another type of strong complexity assumption is to assume RSA is secure against *sub-exponential*-time attackers and apply complexity leveraging techniques. Micali, Rabin and Vadhan [20] did this to construct verifiable unpredictable functions, which immediately admit a signature scheme. In contrast, we only assume the RSA problem is hard for polynomial-time attackers; in other words, all our reductions are polynomial in the security parameter.

Earlier this year, Hohenberger and Waters [18] presented short RSA and CDH based schemes secure in the standard model, where the signer had to maintain a counter value as state. This counter was incremented with each signature issued. Unfortunately, their scheme was compromised if the signer accidentally issued two signatures with the same counter value. Indeed, while early signatures, such as those of Goldwasser, Micali and Rivest [16], were stateful, the concept of the *stateless* signature has become so ingrained in practice that it is really more of a requirement than an extra feature. Moreover, stateful signatures are harder for systems designers to work with because, in addition to protecting the secret key, they must also safeguard a counter value (in writable memory) from being maliciously rolled back by an adversary.

2 Generic Transformation of Weakly-Secure Signatures to Fully-Secure Signatures Using Chameleon Hashes

2.1 Signature Schemes

A signature scheme is a tuple of the following algorithms:

KeyGen(1^λ)**:** the key generation algorithm outputs a keypair (PK, SK).

Sign(SK, M)**:** the signing algorithm takes in a secret key SK, and a message M, and produces a signature σ.

Verify(PK, M, σ)**:** the verification algorithm takes in a public key PK, a message M, and a purported signature σ, and returns 1 if the signature is valid and 0 otherwise.

2.2 GMR Unforgeability

The basic security notion for signatures is *existential unforgeability with respect to adaptive chosen-message attacks* as formalized by Goldwasser, Micali and Rivest [16]. It is defined using the following game between a challenger and an adversary $\mathcal{A}$ over message space $\mathcal{M}$:

Setup: The challenger runs the algorithm **KeyGen**(1^λ) to obtain the public key PK and the secret key SK, and gives PK to the adversary.

Queries: Proceeding adaptively, the adversary may request a signature on any message $M \in \mathcal{M}$ and the challenger will respond with $\sigma \leftarrow \mathbf{Sign}(\mathrm{SK}, M)$. Let Q be the set of messages queried by the adversary.

Output: Eventually, the adversary will output a pair (M, σ) and is said to win the game if $M \notin Q$ and $\mathbf{Verify}(\mathrm{PK}, M, \sigma) = 1$.

We define $\mathbf{Adv}_{\mathcal{A}}$ to be the probability that adversary $\mathcal{A}$ wins in the above game.

Definition 1 (Unforgeability against Adaptive Chosen Message Attacks [16]). *A signature scheme* $(\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify})$ *is existentially unforgeable with respect to adaptive chosen message attacks if for all probabilistic polynomial time adversaries* $\mathcal{A}$*,* $\mathbf{Adv}_{\mathcal{A}}$ *is negligible in* λ*.*

2.3 Weak Unforgeability

Several works (e.g., [3]) consider a weaker definition called *existential unforgeability with respect to weak chosen-message attacks*. It is defined using the following game between a challenger and an adversary $\mathcal{A}$ over message space $\mathcal{M}$:

Queries: The adversary sends the challenger a list Q of messages $M_1, \ldots, M_n \in \mathcal{M}$.

Response: The challenger runs the algorithm $\mathbf{KeyGen}(1^\lambda)$ to obtain the public key PK and the secret key SK. Next, the challenger signs each queried message as $\sigma_i \leftarrow \mathbf{Sign}(\mathrm{SK}, M_i)$ for $i = 1$ to n. The challenger then sends $\mathrm{PK}, \sigma_1, \ldots, \sigma_n$ to the adversary.

Output: Eventually, the adversary will output a pair (M, σ) and is said to win the game if $M \notin Q$ and $\mathbf{Verify}(\mathrm{PK}, M, \sigma) = 1$.

We define $\mathbf{Adv}_{\mathcal{A}}^{weak}$ to be the probability that adversary $\mathcal{A}$ wins in the above game.

Definition 2 (Unforgeability against Weak Chosen Message Attacks). *A signature scheme* $(\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify})$ *is existentially unforgeable with respect to weak chosen message attacks if for all probabilistic polynomial time adversaries* $\mathcal{A}$*,* $\mathbf{Adv}_{\mathcal{A}}^{weak}$ *is negligible in* λ*.*

2.4 Chameleon Hashes

As formalized by Krawczyk and Rabin [19], a chameleon hash function H takes two inputs: a message m and randomness r. It is collision-resistant with the additional property that, given special trapdoor information, any target y and any message m', it is possible to efficiently find a value r' such that $H(m', r') = y$. Secure constructions exist in the standard model under the discrete logarithm assumption [19], the hardness of factoring [19], and the RSA assumption [1,18].

2.5 Generic Transformation

We now recall a generic construction for building unforgeable signatures out of weak unforgeable signatures and chameleon hashes, as used in many prior signature constructions such as [19,28,3,18]. Let (G, S, V) be a weak unforgeable scheme for n-bit messages. Let chameleon hash family $\mathcal{H}$ map inputs as $\{0,1\}^\ell \times \{0,1\}^k \rightarrow \{0,1\}^n$. Consider a scheme for ℓ-bit messages constructed as:

KeyGen(1^λ): Select a random chameleon hash $H \in \mathcal{H}$. Run $G(1^\lambda)$ to obtain the keypair (pk, sk). The public key is $\text{PK} = (pk, H)$ and the secret key is $\text{SK} = (sk, H)$.

Sign($\text{SK}, M \in \{0,1\}^\ell$): Pick a random $r \in \{0,1\}^k$. Compute $x = H(M, r)$, and then $\sigma' \leftarrow S(sk, x)$. Output the signature $\sigma = (\sigma', r)$.

Verify(PK, M, σ): Parse σ as (σ', r). Compute $x = H(M, r)$ and then output $V(pk, x, \sigma')$.

Lemma 1. *If (G, S, V) is a weakly-secure scheme according to Definition 2 and $\mathcal{H}$ is a secure chameleon hash family, then the above scheme is a fully-secure scheme according to Definition 1.*

While this construction is well known (e.g., [19,28,3,18]), we provide an explicit proof of the above lemma in the full version of this work.

3 Algebraic Settings and Complexity Assumptions

3.1 RSA Assumption and Other Facts

We begin by recalling some basic facts and complexity assumptions.

Assumption 1 (RSA [25]). *Let k be the security parameter. Let positive integer N be the product of two k-bit, distinct odd primes p, q. Let e be a randomly chosen positive integer less than and relatively prime to $\phi(N) = (p-1)(q-1)$. Given (N, e) and a random $y \in \mathbb{Z}_N^*$, it is hard to compute x such that $x^e \equiv y \mod N$.*

In the **Strong RSA** assumption, the adversary is given (N, y) and succeeds by producing any integer pair (e, x) such that $e > 1$ and $x^e \equiv y \mod N$. The standard RSA version is much more restrictive on the adversary.

In Section 4, we will restrict ourselves to the RSA assumption where $N = pq$ is the product of two safe primes $p = 2p' + 1$ and $q = 2q' + 1$. (Technically, we will want that the prime exponents used during signing do not divide $\phi(N)$. While safe primes will make this argument simpler, they are not necessary.)

Our RSA-based scheme will require a primality test, such as the efficient test of Miller and Rabin [21,24]. We will also use the following facts.

Lemma 2 (Shamir [27]). *Given $x, y \in \mathbb{Z}_n$ together with $a, b \in \mathbb{Z}$ such that $x^a = y^b$ and $\gcd(a, b) = 1$, there is an efficient algorithm for computing $z \in \mathbb{Z}_n$ such that $z^a = y$.*

Theorem 2 (Prime Number Theorem). *Define $\pi(x)$ as the number of primes $\leq x$. For $x > 1$,*

$$\pi(x) > \frac{x}{\lg x}.$$

3.2 Bilinear Groups and the CDH Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p. A *bilinear map* is an efficient mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ which is both: (*bilinear*) for all $g \in \mathbb{G}$ and $a, b \leftarrow \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; and (*non-degenerate*) if g generates $\mathbb{G}$, then $e(g, g) \neq 1$.

Assumption 3 (Computational Diffie-Hellman [8]). *Let g generate a group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr[a, b, \leftarrow \mathbb{Z}_p; z \leftarrow \mathcal{A}(g, g^a, g^b) : z = g^{ab}].$$

4 An RSA-Based Construction

4.1 A Weakly-Secure Scheme

Setup(1^λ). The setup algorithm chooses an RSA modulus N, such that $2^\ell < \phi(N) < 2^{\ell+2}$, where ℓ is another security parameter derived from 1^λ. It then chooses a random value $h \in \mathbb{Z}_N^*$.

Next, it chooses a random key K for the PRF function $F : \{0,1\}^* \rightarrow \{0,1\}^\ell$ and a random $c \in \{0,1\}^\ell$. It then establishes a function $H_{(\cdot)} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ as follows:

$$H_{K,c}(z) = F_K(i, z) \oplus c,$$

where i, called the *resolving index* for z, is the smallest $i \geq 1$ such that $F_K(i, z) \oplus c$ is odd and prime.

The public key PK is (N, h, c, K), where anyone can compute $H()$ using c and K from the public key. The secret key SK is the factorization of N together with the (public) values (c, K), which are necessary for the signer to compute $H()$.

Sign$(\mathrm{SK}, M \in \{0,1\}^n)$. To sign messages larger than n bits, one could first apply a collision-resistant hash function to the message. Let $M^{(i)}$ denote the first i bits of M; that is, the length i prefix of M. For $i = 1$ to n, it computes $e_i = H_{K,c}(M^{(i)})$. Finally, it outputs the signature

$$\sigma = h^{\prod_{i=1}^n e_i^{-1}} \mod N.$$

Note: if any e_i divides $\phi(N)$, then σ may not be defined. In this event, the signer will output SK as the signature, since we are using safe primes and thus $2e_i + 1$ divides N. We will later argue that this event occurs with negligible probability.

Verify(PK, M, σ). The verification algorithm first computes the appropriate primes as follows: for $i = 1$ to n, it computes $e_i = H_{K,c}(M^{(i)})$. The algorithm accepts if and only if

$$\sigma^{\prod_{i=1}^{n} e_i} \equiv h \mod N.$$

4.2 Proof of Security

Theorem 4 (Weak Security under RSA). *If the RSA assumption holds when N is the product of two safe primes, then the above signature scheme is weakly unforgeable as in Definition 2.*

Proof. As in the stateful signatures of [18], our reduction will disregard all RSA challenges (N, e^*, y) where e^* is *not* an odd *prime* less than 2^ℓ. We recall from [18] that good challenges will occur with polynomial probability. By construction, $\phi(N) < 2^{\ell+2}$. We also know, by Theorem 2, that the number of primes $\leq 2^\ell$ is $\geq \frac{2^\ell}{\ell}$. Thus, a loose bound on the probability of e^* being a prime in the proper range is $(\frac{2^\ell}{\ell})/2^{\ell+2} = \frac{1}{4\ell}$.

Suppose there is an adversary $\mathcal{A}$ against the above signature scheme for n-bit messages that makes at most $q(\lambda)$ queries where $q()$ is a polynomial and succeeds in forging with probability ϵ. (We say q queries where it is clear from context.) We show that this adversary can be used to break (good challenges for) RSA with probability approximately $\epsilon/(qn\ell\lambda)$, where q, n, ℓ are all polynomial in the security parameter λ. On input (N, e^*, y), where e^* is an odd prime $< 2^\ell$, our RSA solver $\mathcal{B}$ proceeds as:

Setup: Adversary $\mathcal{A}$ must first provide $\mathcal{B}$ with the messages $M_1, \ldots, M_q$ on which it will request to see signatures. $\mathcal{B}$ wishes to guess the shortest prefix of M^*, the message on which the adversary will later forge, that is different from all other prefixes of $M_1, \ldots, M_q$.

– If $q = 0$, $\mathcal{B}$ guesses $w \in \{0, 1\}$ at random and sets value $t^* = 1$. When $\mathcal{A}$ does not ask for any signatures, then the first prefix (i.e., bit) of the forgery message M^* will be used later to embed the challenge, and $\mathcal{B}$ need only guess it with probability 1/2.
– If $q \geq 1$, the simulator guesses at random $1 \leq i^* \leq q$ (a message with the longest prefix in common with the forgery message[1]) and $1 \leq t^* \leq n$ (the length of the longest common prefix plus one). We will later argue that $\mathcal{B}$'s guesses are correct with probability $\geq 1/(qn)$. The values (i^*, t^*) define the t^*-bit string w comprised of the first $(t^* - 1)$ bits of M_{i^*} followed by the complement of M_{i^*}'s t^* bit. In other words, if $\mathcal{B}$'s guesses are correct, then we know that w is the t^*-bit prefix of the message on which the adversary will forge, and moreover, that no other signatures will be issued with this prefix.

[1] More than one message in $M_1, \ldots, M_q$ may share this longest common prefix. Guessing any one of them will suffice for this analysis.

Armed with this information, $\mathcal{B}$ proceeds to set up the public key as:

1. Select a random PRF seed K.
2. Select a random index $1 \le j \le \ell\lambda$ and set $c = F_K(j, w) \oplus e^*$.
3. Abort if any of the following conditions hold:
   (a) j is not the resolving index of $H_{K,c}(w)$.
   (b) Some prime is not locally unique or divides $\phi(N)$. Let $P(M_i)$ be the vector of n primes derived as $H_{K,c}(M_i^{(k)})$ for $k = 1$ to n. Abort if, for any i, $P(M_i)$ contains a repeated prime or a prime that divides $\phi(N)$ (i.e., a prime p such that $2p + 1$ divides N).
   (c) $e^* \in S$, where S is defined as the set of all unique primes across all vectors $P(M_i)$ for $i = 1$ to q.
4. Set
$$h = y^{\prod_{e_i \in S} e_i} \mod N.$$
   The maximum size of S is $qn - 1$. To $\mathcal{A}$, h will appear to be distributed randomly in $\mathbb{Z}_N^*$.
5. Send the public key $\mathrm{PK} = (N, h, c, K)$ to $\mathcal{A}$.

Sign: $\mathcal{B}$ can create a signature on any message M provided during the Setup as follows.

1. Compute the vector of n primes $P(M)$ (i.e., the set $H_{K,c}(M_i^{(k)})$ for $k = 1$ to n).
2. Compute the signature as
$$\sigma = y^{\prod_{e_i \in [S - P(M)]} e_i} \mod N.$$

Extract from Forgery: Eventually, $\mathcal{A}$ will output a forgery (M^*, σ). If $M^{*(t^*)} \ne w$, then abort; the Setup guess was not correct.

Now, we wish to extract the RSA solution. Consider the vector of primes $P(M^*)$. If any member of $P(M^*)$ divides $\phi(N)$ (i.e., a prime p such that $2p+1$ divides N), then $\mathcal{B}$ can factor N and compute the RSA solution $y^{1/e^*} \mod N$.

Otherwise, let α be the number of times e^* appears in $P(M^*)$. We know from our Setup that $\alpha \ge 1$. Now, consider the following settings:

$$x = \sigma^{(e^*)^{\alpha-1} \prod_{e_i \in P(M^*), e_i \ne e^*} e_i}, \quad y = y, \quad a = e^*, \quad b = \prod_{e_i \in S} e_i$$

First, we see that $x^a = y^b$. Second, we know that $\gcd(a, b) = 1$, since all values are primes and $e^* \notin S$. Thus, $\mathcal{B}$ can apply Lemma 2 to efficiently compute a value $z \in \mathbb{Z}_N$ such that $z^a = y$. $\mathcal{B}$ outputs z as the RSA solution.

Analysis. We now argue that any successful adversary $\mathcal{A}$ against our scheme will have success in the game presented by $\mathcal{B}$. To do this, we first define a sequence of games, where the first game models the real security game and the final game is exactly the view of the adversary when interacting with $\mathcal{B}$. We then show via a series of claims that if a $\mathcal{A}$ is successful against Game j, then it will also be successful against Game $j + 1$.

Game 1: This game is defined to be the same as the security game of the scheme.

Game 2: The same as Game 1, with the exception that $\mathcal{A}$ fails if some prime is not locally unique or divides $\phi(N)$ (as described in Setup abort condition (b)).

Game 3: The same as Game 2, with the exception that $\mathcal{A}$ fails if $e^* \in S$.

Game 4: The same as Game 3, with the exception that at the beginning of the game $\mathcal{B}$ guesses w as follows:

- if $q = 0$, w is chosen at random from $\{0, 1\}$;
- else, a random pair (i^*, t^*) is chosen, where $1 \le i^* \le q$ and $1 \le t^* \le n$. Together with $M_1, \ldots, M_q$, this defines the string w as comprised of the first $(t^* - 1)$ bits of M_{i^*} followed by the complement of M_{i^*}'s t^*th bit.

Now $\mathcal{A}$ fails if the message on which he forges does not have prefix w.

Game 5: The same as Game 4, with the exception that $\mathcal{A}$ fails if the resolving index of $H_{K,c}(w)$ is greater than $\ell\lambda$.

Game 6: The same as Game 5, with the exception that at the beginning of the game $\mathcal{B}$ guesses an index $1 \le j^* \le \ell\lambda$ and $\mathcal{A}$ fails is the resolving index of $H_{K,c}(w)$ is not j^*.

Game 7: The same as Game 6, with the exception that at the beginning of the game $\mathcal{B}$ chooses a random PRF seed K (as before) and a random $e \in \{0, 1\}^\ell$ and then sets $c = F_K(j^*, w) \oplus e$.

Game 8: The same as Game 7, with the exception that c is set as $c = F_K(j^*, w) \oplus e^*$, where e^* is the ℓ-bit prime from the RSA challenge.

Game 8 is exactly the view of the adversary when interacting with $\mathcal{B}$. In the full version of this paper, we complete this argument by linking the probability of $\mathcal{A}$'s success in these games via a series of claims. The only non-negligible probability gaps come between Games 3 and 4, where there is a factor $1/(qn)$ loss, and between Games 5 and 6, where there is a factor $1/(\ell\lambda)$ loss.

4.3 Short, Fully-Secure RSA Signatures

We obtain a fully-secure signature scheme by combining our RSA-based weakly unforgeable signatures with any suitable chameleon hash function. Standard model chameleon hashes exist under the hardness of factoring [19] and RSA [1,18]. The following result is immediate from Theorem 4 and Lemma 1.

Corollary 1 (Full Security under RSA). *Let (G', S', V') be the signature scheme described in Section 4.1. Let $\mathcal{H}$ be a chameleon hash function family secure under the RSA assumption. Let (G, S, V) be the signature scheme resulting from the generic transformation in Section 2.5 on (G', S', V') and $\mathcal{H}$. Then (G, S, V) is a fully-secure signature scheme, according to Definition 1, under the RSA assumption.*

The resulting signatures are very short. A signature contains one element from $\mathbb{Z}_N^*$ and one k-bit integer, where k is derived from the security parameter and the settings of the chameleon hash when using the standard model, RSA-based hash in [1,18]. We provide more details on this in the full version.

5 Optimizations for the RSA Construction

While the main efficiency focus of this work is on simultaneously achieving a short public key and signature under RSA, we now briefly turn our attention to methods for improving the computational efficiency of these signatures. A significant computational overhead for both the signer and the verifier in our RSA scheme is the generation and testing of primes necessary to compute the hash function $H()$. The signer also must perform one exponentiation, where the exponent may be reduced modulo $\phi(N)$, while the verification cost is roughly n exponentiations of ℓ-bit exponents.

5.1 Online/Offline Signatures

In an *online/offline* signature as introduced by Even, Goldreich and Micali [11], the scheme is designed so that the signer can do the bulk of his computational work *before* the message is known to him. This paradigm is extremely useful for many applications which require a quick response time once a message comes in, but where the device may otherwise have relatively longer periods of inactivity. Fortunately, our RSA scheme (as well as our later CDH scheme) have this property.

To see this, recall the generic structure of our fully-secure signature scheme from Section 2.5. The signer can, offline, choose a random n-bit message X, sign X using the weakly-secure scheme, and then later use the trapdoor of the chameleon hash to link this signature to any desired message M. Thus, all of the expensive primality testing and the single exponentiation for our scheme can be performed offline by the signer. Indeed, this use of a chameleon hash to obtain online/offline signatures was previously suggested by Shamir and Tauman [28].

5.2 Send Resolving Indices with the Signature

One of the main verification costs is searching for the n resolving indices. Each signature verification requires an expected $n\ell$ primality tests; i.e., an expected ℓ per evaluation of $H_{K,c}(M^{(i)})$, for $i = 1$ to n. The number of primality tests could be reduced to n by sending a vector of resolving indices along with the signature. While the size of the signature would increase by roughly $n \cdot \log(\ell\lambda)$ bits (i.e., the number of resolving indices n by the representation of their maximum likely value $\ell\lambda$), this is still considerably smaller than several prior tree-based approaches.

The danger of attack on this scheme is that a malicious signer will send a higher index than the resolving index. However, suppose that a maximum resolving index $T = \ell\lambda$ is posted in the public key and that honest verifiers reject if any of the sender's indices exceed this value. Then we can alter our prior proof to fit this variation as well. The simulator $\mathcal{B}$ behaves as before, except that she guesses which resolving index the adversary $\mathcal{A}$ will choose for the evaluation of w, between 1 and $T = \ell\lambda$, and later uses this value to extract the RSA solution. As $\mathcal{B}$ is already guessing the resolving index in our current reduction (see Game 6), there is no additional concrete security loss.

5.3 Using a Larger Alphabet

In the current scheme, the message is broken up into n 1-bit chunks, which each correspond to a prime exponent. Finding these n primes is costly for both the signer and the verifier, and the verification then requires n exponentiations (of one prime each). Suppose we break the message up into larger, k-bit chunks. The benefit of this approach would be that only n/k primes need now be found and used in verification. The drawback is that the concrete security of the scheme would decrease by a factor of $1/(2^k - 1)$, because now the simulator must guess within the chunk pointed to by (i^*, t^*), which of the $2^k - 1$ values the forger will later use. In the binary case, the bit pointed to by (i^*, t^*) was always the complement of M_{i^*}'s t^*th bit.

Considering $k = 2$, however, we cut the cost of signature generation and verification in half, for only a 1/3 reduction in concrete security. In some scenarios, this may be acceptable.

5.4 Using Smaller Prime Exponents

In the current scheme, primes are chosen to be of ℓ bits where 2^ℓ is roughly the size of $\phi(N)$. We could instead select ℓ to be smaller, but where 2^ℓ is still exponential in the security parameter. The main benefit of this approach would be a slightly more efficient scheme at the cost of a security proof with respect to a different variant of the RSA problem, namely, inverting RSA with a random prime exponent of bit-length less than or equal to ℓ.

6 A CDH-Based Construction

Our RSA proof techniques can be translated into the CDH setting as well. Interestingly, this provides new insights about the security of the only prior (stateless) scheme known to be secure under CDH in the standard model: the Waters signatures [29]. We present an entirely new method for reasoning about the *weak* unforgeability of these signatures under CDH. By adding a chameleon hash, we obtain a fully-secure scheme which is a derivative of the Waters signatures. The main contribution here is a much shorter, cleaner security argument as well as a demonstration that our RSA proof techniques are likely to be useful in other settings.

6.1 The Waters Signatures

Recall the Waters signature scheme [29], which is known to be fully-secure under the Computational Diffie-Hellman assumption in the standard model.

Setup(1^λ). The setup algorithm selects a bilinear group $\mathbb{G}$ of prime order $p > 2^\lambda$. It chooses a random exponent $a \in \mathbb{Z}_p$. Let n be a security parameter derived from λ. It chooses random group elements $g, v_0, v_1, \ldots, v_n \in \mathbb{G}$. The secret key is a and the public key is output as:

$$g, v_0, v_1, \ldots, v_n, e(g, g)^a.$$

Sign(SK, $M \in \{0,1\}^n$). The message space is treated as n-bits; to sign arbitrarily long messages one could first apply a collision-resistant hash function. Here M_i denotes the ith bit of M. The signer chooses a random $r \in \mathbb{Z}_p$ and then outputs the signature as:

$$\sigma_1 = g^a \left(v_0 \prod_{i=1}^{n} v_i^{M_i} \right)^r, \quad \sigma_2 = g^r.$$

Verify(PK, $M \in \{0,1\}^n, \sigma = (\sigma_1, \sigma_2)$) The verification algorithm uses the bilinear map to verify the signature by checking that

$$e(\sigma_1, g) = e(g,g)^a e(v_0 \prod_{i=1}^{n} v_i^{M_i}, \sigma_2).$$

6.2 Proof of Security

Theorem 5 (Weak Security under CDH). *If the CDH assumption holds in $\mathbb{G}$, then the Waters signature scheme is weakly unforgeable as in Definition 2.*

Proof. Suppose we have an adversary $\mathcal{A}$ against the above signature scheme that makes at most $q(\lambda)$ queries where $q()$ is a polynomial and succeeds in forging with probability ϵ. (We say q queries where it is clear from context.) We show that this adversary can be used to break CDH with probability $\geq \epsilon/(qn)$. On input (g, g^a, g^b), our CDH solver $\mathcal{B}$ proceeds as follows:

Setup: Adversary $\mathcal{A}$ must first provide $\mathcal{B}$ with the messages $M_1, \ldots, M_q$ on which it will request to see signatures. $\mathcal{B}$ wishes to guess the shortest prefix of M^*, the message on which the adversary will later forge, that is different from all other prefixes of $M_1, \ldots, M_q$.

- If $q = 0$, $\mathcal{B}$ guesses $w \in \{0,1\}$ at random and sets value $t^* = 1$. When $\mathcal{A}$ does not ask for any signatures, then the first prefix (i.e., bit) of the forgery message M^* will be used later to embed the challenge, and $\mathcal{B}$ need only guess it with probability 1/2.
- If $q \geq 1$, the simulator guesses at random $1 \leq i^* \leq q$ (a message with the longest prefix in common with the forgery message[2]) and $1 \leq t^* \leq n$ (the length of the longest common prefix plus one). We will later argue that $\mathcal{B}$'s guesses are correct with probability $\geq 1/(qn)$. The values (i^*, t^*) define the t^*-bit string w comprised of the first $(t^* - 1)$ bits of M_{i^*} followed by the complement of M_{i^*}'s t^* bit. In other words, if $\mathcal{B}$'s guesses are correct, then we know that w is the t^*-bit prefix of the message on which the adversary will forge, and moreover, that no other signatures will be issued with this prefix.

[2] More than one message in $M_1, \ldots, M_q$ may share this longest common prefix. Guessing any one of them will suffice for this analysis.

Armed with this information, $\mathcal{B}$ proceeds to set up the public key as:

1. Set $e(g,g)^\alpha = e(g^a, g^b)$, thus the secret key will implicitly be set to $\alpha = ab$.
2. Pick random values $y_0, \ldots, y_n \in \mathbb{Z}_p$.
3. Set $v_0 = g^{y_0} \prod_{i=1}^{t^*} (g^a)^{w_i}$.
4. For $i = 1$ to n, set

$$v_i = \begin{cases} g^{y_i} & \text{if } i > t^*; \\ g^{-a} g^{y_i} & \text{else if } w_i = 1; \\ g^a g^{y_i} & \text{otherwise } (w_i = 0). \end{cases}$$

   Here w_i denotes the ith bit of w. The key observation here is that all g^a terms will cancel out for signatures with prefix w and that this won't be true for any other t^*-bit prefix.
5. Send $\mathrm{PK} = (g, v_0, \ldots, v_n, e(g,g)^\alpha)$ to $\mathcal{A}$.

Sign: $\mathcal{B}$ can create a signature on any message M provided during the Setup. Let $\beta = \sum_{i=1}^{t^*} w_i$ be the number of 1's in w. Let $\gamma = \sum_{i=1}^{t^*} m_i(1 - 2w_i)$, where m_i denotes the ith bit of M. Notice that $\beta + \gamma = \sum_{i=1}^{t^*} w_i + m_i(1-2w_i)$; this is equal to the number of bits that differ between w and the first t^* bits of M. By our setup, $\beta + \gamma \neq 0$ for all messages provided by the adversary.

1. Select a random value $r' \in \mathbb{Z}_p$.
2. Set $\sigma_2 = (g^{-b})^{1/(\beta+\gamma)} g^{r'}$; this implicitly sets $\sigma_2 = g^r$ with $r = -b/(\beta+\gamma) + r'$.
3. Set $\sigma_1 = \sigma_2^{y_0 + \sum_{i=1}^n m_i y_i} g^{ar'(\beta+\gamma)}$. To see that this is properly formed relative to σ_2, note that the value we want is:

$$\sigma_1 = g^{ab} \left(v_0 \prod_{i=1}^n v_i^{m_i} \right)^r = g^{ab} \left(g^{y_0 + \beta a} g^{\gamma a} g^{\sum_{i=1}^n m_i y_i} \right)^r =$$
$$g^{ab} \left(g^{a(\beta+\gamma)} \right)^r \left(g^{y_0 \sum_{i=1}^n m_i y_i} \right)^r = \left(g^{a(\beta+\gamma)} \right)^{r'} \left(g^{y_0 \sum_{i=1}^n m_i y_i} \right)^r =$$
$$\sigma_2^{y_0 + \sum_{i=1}^n m_i y_i} g^{ar'(\beta+\gamma)}.$$

4. Output the signature (σ_1, σ_2).

Extract from Forgery: Eventually, $\mathcal{A}$ will output a forgery $(M, \sigma = (\sigma_1, \sigma_2))$. If $M^{(t^*)} \neq w$, then abort; the Setup guess was not correct. From the construction, one can see that $\mathcal{B}$'s guesses are correct with probability $\geq 1/(qn)$, because the distribution of the public key and signature responses is the same for all possible guesses. Now, to extract the CDH solution g^{ab}, the main idea is that the forgery is of the form $\sigma_1 = g^{ab} g^{zr}, \sigma_2 = g^r$ for a value z known to $\mathcal{B}$, and thus it can compute $\sigma_1/\sigma_2^z = g^{ab}$. To see this, let m_i denote the ith bit of M and observe that:

$$g^z = v_0 \prod_{i=1}^n v_i^{m_i} = g^{y_0 + \beta a} \prod_{i=1}^{t^*} g^{a m_i (1 - 2w_i)} \prod_{i=1}^n g^{m_i y_i},$$

and thus that

$$z = y_0 + \left(\beta + \sum_{i=1}^{t^*} m_i(1 - 2w_i)\right) a + \sum_{i=1}^{n} m_i y_i.$$

Observe that up to t^*, it holds that $m_i = w_i$. Recall that $\beta = \sum_{i=1}^{t^*} w_i$ was chosen as the number of 1's in w. Thus, $\beta + \sum_{i=1}^{t^*} w_i(1 - 2w_i) = \beta - \sum_{i=1}^{t^*} w_i$ is equal to zero and z simplifies to $y_0 + \prod_{i=1}^{n} g^{m_i y_i}$.

6.3 Short, Fully-Secure CDH Signatures

We obtain a fully-secure signature scheme by combining the above CDH-based weakly unforgeable signatures with any suitable chameleon hash function. Standard model chameleon hashes exist under the discrete-logarithm assumption [19] (and thus under CDH). The following result is immediate from Theorem 5 and Lemma 1.

Corollary 2 (Full Security under CDH). *Let (G', S', V') be the signature scheme described in Section 6.1. Let $\mathcal{H}$ be a chameleon hash function family secure under the CDH assumption. Let (G, S, V) be the signature scheme resulting from the generic transformation in Section 2.5 on (G', S', V') and $\mathcal{H}$. Then (G, S, V) is a fully-secure signature scheme, according to Definition 1, under the CDH assumption.*

The resulting signatures are fairly short. A signature contains two elements from $\mathbb{G}$ and one k-bit value, where k is derived from the security parameter and the choice of the chameleon hash. Weak signing requires only two exponentiations, since the signer can choose $v_0, \ldots, v_n$ such that she knows their discrete logarithms base g. Verification requires only two pairings. Of course, this is mostly a theoretical exercise as the Waters signatures are more efficient on all counts.

7 Conclusion and Open Problems

In this work, we presented the first stateless signatures with short public keys and short signatures secure under the RSA assumption in the standard model. This answers a long-standing open problem as to whether or not such a construction was possible. Indeed, this is the only known scheme to satisfy all of the above requirements under a computational assumption with a short input and a single valid output.

The construction requires a new proof technique for reasoning about the security of signature schemes. We demonstrate that this technique is of broader interest by showing how to apply it in the CDH setting to obtain a new security proof for the Waters signatures [29]. Interestingly, both our constructions are also online/offline signatures, where the vast majority of the signing computation can be done offline before the signer knows the message.

We leave open several interesting problems. The Waters signatures and our variant here offer short signatures, but a public key of $O(\lambda)$ elements, where λ is the security parameter. It is still unknown how to realize standard model CDH signatures where both the signatures and the public key are short. While we offer many computational optimizations for our RSA scheme in Section 5, it would be of great practical significance to obtain faster signing and verification times. Finally, given the usefulness of signatures in designing stronger encryption, anonymous credentials, electronic cash, etc., it would be worthwhile to revisit some of these systems and try to weaken the complexity assumptions on which they are founded.

Acknowledgments

We thank Amit Sahai for helpful discussions and the anonymous reviewers for their helpful comments.

References

1. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security (CCS), pp. 62–73. ACM Press, New York (1993)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
5. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
6. Cramer, R., Damgård, I.: New generation of secure and practical RSA-based signatures. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 173–185. Springer, Heidelberg (1996)
7. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. ACM Transactions on Information and System Security 3(3), 161–185 (2000)
8. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
9. Dwork, C., Naor, M.: An efficient existentially unforgeable signature scheme and its applications. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 234–246. Springer, Heidelberg (1994)
10. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
11. Even, S., Goldreich, O., Micali, S.: On-line/off-line digital schemes. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)

12. Fischlin, M.: The Cramer-Shoup Strong-RSA signature scheme revisited. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
13. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Symposium on the Theory of Computing (STOC), pp. 197–206 (2008)
15. Goh, E.-J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. J. of Cryptology 20(4), 493–514 (2007)
16. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Computing 17(2) (1988)
17. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
18. Hohenberger, S., Waters, B.: Realizing hash-and-sign signatures under standard assumptions. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 333–350. Springer, Heidelberg (2009)
19. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Network and Distributed System Security Symposium (2000)
20. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: Symposium on Foundations of Computer Science (FOCS), pp. 120–130. IEEE Computer Society Press, Los Alamitos (1999)
21. Miller, G.L.: Riemann's hypothesis and tests for primality. Journal of Computer and System Sciences 13, 300–317 (1976)
22. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
23. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
24. Rabin, M.O.: Probabilistic algorithm for testing primality. Journal of Number Theory 12, 128–138 (1980)
25. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Comm. of the ACM 21(2), 120–126 (1978)
26. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)
27. Shamir, A.: On the generation of cryptographically strong pseudorandom sequences. ACM Transaction on Computer Systems 1, 38–44 (1983)
28. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
29. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Smooth Projective Hashing for Conditionally Extractable Commitments

Michel Abdalla, Céline Chevalier, and David Pointcheval

École Normale Supérieure, CNRS-INRIA, Paris, France

Abstract. The notion of smooth projective hash functions was proposed by Cramer and Shoup and can be seen as special type of zero-knowledge proof system for a language. Though originally used as a means to build efficient chosen-ciphertext secure public-key encryption schemes, some variations of the Cramer-Shoup smooth projective hash functions also found applications in several other contexts, such as password-based authenticated key exchange and oblivious transfer. In this paper, we first address the problem of building smooth projective hash functions for more complex languages. More precisely, we show how to build such functions for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which smooth projective hash functions are known to exist. Next, we illustrate how the use of smooth projective hash functions with more complex languages can be efficiently associated to extractable commitment schemes and avoid the need for zero-knowledge proofs. Finally, we explain how to apply these results to provide more efficient solutions to two well-known cryptographic problems: a public-key certification which guarantees the knowledge of the private key by the user without random oracles or zero-knowledge proofs and adaptive security for password-based authenticated key exchange protocols in the universal composability framework with erasures.

1 Introduction

In [16], Cramer and Shoup introduced a new primitive called smooth projective hashing and showed how to use it to generalize their chosen-ciphertext secure public-key encryption scheme [15]. The new abstraction not only provided a more intuitive description of the original encryption scheme, but also resulted in several new instantiations based on different security assumptions such as quadratic residuosity and N-residuosity [31].

The notion of smooth projective hash functions (SPHF, [16], after slight modifications [22]) has been proven quite useful and has found applications in several other contexts, such as password-based authenticated key exchange (PAKE, [22]) and oblivious transfer [27]. In the context of PAKE protocols, the work of Gennaro and Lindell abstracted and generalized (under various indistinguishability assumptions) the earlier protocol by Katz, Ostrovsky, and Yung [28] and has become the basis of several other schemes [1,3,8]. In the context of oblivious transfer, the work of Kalai [27] also generalized earlier protocols by Naor and Pinkas [30] and by Aiello, Ishai, and Reingold [2].

To better understand the power of SPHF, let us briefly recall what they are. First, the definition of SPHF requires the existence of a domain X and an underlying NP language L such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. For instance, in the particular case of the PAKE scheme in [13], the language L is defined as the set of triples $\{(c, \ell, m)\}$ such that c is an encryption of m with label ℓ under a public key given in the common reference string (CRS). The semantic security of the encryption scheme guarantees computational indistinguishability between elements from L and from X.

One of the key properties that make SPHF so useful is that, for a point $x \in L$, the hash value can be computed using either a *secret* hashing key hk, or a *public* projected key hp (depending on x [22] or not [16]) together with a witness w to the fact that $x \in L$. Another important property of these functions is that, given the projected key hp, their output is uniquely defined for points $x \in L$ and statistically indistinguishable from random for points $x \in X \setminus L$. Moreover, without the knowledge of the witness w to the fact that $x \in L$, the output of these functions on x is also pseudo-random.

The first main contribution of this paper is to extend the line of work on SPHF, the element-based version proposed by [22], to take into account more complex NP languages. We show how to build SPHF for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which SPHF are known to exist. For instance, let H_m represent a family of SPHF for the language $\{(c)\}$, where c is the encryption of m under a given public key. Using our tools, one can build a family of SPHF for the language $\{(c)\}$, where c is the encryption of either 0 or 1, by combining H_0 and H_1.

One of the advantages of building SPHF for more complex languages is that it allows us to simplify the design of the primitives to which they are associated. To demonstrate this, we consider in this paper the specific case of extractable commitment schemes. In most protocols in which extractable commitments are used, the capability of extracting the committed message usually depends on the commitment being properly generated. To achieve this goal and enforce the correct generation of the commitment, it is often the case that additional mechanisms, such as zero-knowledge proofs, may have to be used. This is the case, for instance, of several protocols where a specific public-key registration phase is required, such as most of the cryptographic protocols with dynamic groups (multisignatures [9,29], group signatures [18], etc). Such a framework is sometimes named *registered public-key model*, where a proof of knowledge of the secret key is required before any certification.

To be able to build more efficient extractable commitment schemes and avoid the use of possibly expensive concurrent zero-knowledge proofs, a second main contribution of this paper is to generalize the concept of extractable commitments so that extraction may fail if the commitment is not properly generated. More specifically, we introduce a new notion of L-extractable commitments in which extraction is only guaranteed if the committed value belongs to the language L and may fail otherwise. The main intuition behind this generalization is that, when used together with a SPHF for the language L, the cases in which extraction may fail will not be very important as the output of the SPHF will be statistically indistinguishable from random in such cases.

Applications

Registered Public-Key Setting. For many cryptographic protocols, for proving the security even when users can dynamically join the system, the simulator described in the security proof often needs to know the private keys of the authorized users, which is called the *registered public-key setting*, in order to avoid rogue-attacks [9]. This should anyway be the correct way to proceed for a certification authority: it certifies a public key to a user if and only if the latter provides a proof of knowledge of the associated private key. However, in order to allow concurrency, intricate zero-knowledge proofs are required, which makes the certification process either secure in the random oracle model [6] only, or inefficient in the standard model.

In this paper, we show how SPHF with conditionally extractable commitments can help to solve this problem efficiently, in the standard model, by establishing a secure channel between the players, with keys that are either the same for the two parties if the commitment has been correctly built, or perfectly independent in the other case.

Adaptively-secure PAKE schemes. We thereafter study more involved key exchange schemes. In 1992, Bellovin and Merritt [7] suggested a method to authenticate a key exchange based on simple passwords, possibly drawn from a space so small that an adversary might enumerate off-line all possible values. Because of the practical interest of such a primitive, many schemes have been proposed and studied. In 2005, Canetti *et al.* [13] proposed an ideal functionality for PAKE protocols, in the universal composability (UC) framework [11,14], and showed how a simple variant of the Gennaro-Lindell methodology [22] could lead to a secure protocol. Though quite efficient, their protocol is not known to be secure against adaptive adversaries, where they can corrupt players at any time, and learn their internal states. The first ones to propose an adaptively-secure PAKE in the UC framework were Barak *et al.* [3] using general techniques from multi-party computation (MPC). Though conceptually simple, their solution yields quite inefficient schemes.

Here, we take a different approach. Instead of using general MPC techniques, we extend the Gennaro-Lindell methodology to deal with adaptive corruptions by using a non-malleable conditionally-extractable and equivocable commitment scheme with an associated SPHF family. The new scheme is adaptively secure in the common reference string model in the UC framework under standard complexity assumptions with erasures.

Related work

Commitments. Commitment schemes are one of the most fundamental cryptographic primitives, being used in several cryptographic applications such as zero-knowledge proofs [25] and secure multi-party computation [24]. Even quite practical protocols need them, as already explained above in the public-key registration setting, but also in password-based authenticated key exchange [22].

They allow a user to commit a value x into a public value C, such that the latter does not reveal any information about x (the hiding property), but C can be opened later to x only: one cannot change its mind (the binding property). Various additional properties are often required, such as non-malleability, extractability and equivocability. Canetti and Fischlin [12] provided an ideal functionality for such a primitive and showed that achieving all these properties at the same time was impossible in the UC plain model. They also provided the first candidate in the CRS model. Damgård and Nielsen [17] later proposed another construction of universally composable commitments, that is more efficient for some applications. Since we want to avoid the use of possibly inefficient proofs of relations present in the Damgård-Nielsen construction and given that the Canetti-Fischlin construction is well suited for our purpose of designing an associated smooth hash function, we opted to use the latter as the starting point for our constructions.

PAKE. The password-based setting was first considered by Bellovin and Merritt [7] and followed by many proposals. In 2000, Bellare, Pointcheval, and Rogaway [5] as well as Boyko, MacKenzie, and Patel [10] proposed security models and proved variants of the protocol of [7], under ideal assumptions, such as the random oracle model [6]. Soon after, Katz, Ostrovsky, and Yung [28] and Goldreich and Lindell [23] proposed the first protocols with a proof of security in the standard model, with the former being based on the decisional Diffie-Hellman assumption and the latter on general assumptions. Later, Gennaro and Lindell [22] proposed an abstraction and generalization of the KOY protocol and became the basis of several other variants, including ours in the last section.

Organization of the Paper. In Section 2, we review the basic primitives needed in this paper. Then, in Section 3, we describe our first contribution: SPHF families on conjunctions and disjunctions of languages. In Section 4 we combine that with our second contribution, conditionally-extractable commitments. We focus on the ElGamal-based commitment, since this is enough to build more efficient public-key certification protocols. Finally, in Section 5, we add equivocability to the commitment, borrowing techniques from Canetti and Fischlin [12]. Then, we add the non-malleability property, granted the Cramer-Shoup encryption scheme, which can then be used to build an adaptively-secure PAKE in the UC framework, based on the Gennaro and Lindell [22] framework. Due to space restrictions, formal definitions, proofs, and application details were postponed to the appendix.

2 Commitments

In the following, we focus on Pedersen commitments, and certification of Schnorr-like public keys, hence, we work in the discrete logarithm setting. As a consequence, to get extractable commitments, we use encryption schemes from the same family: the ElGamal encryption [21] and the labeled version of the Cramer-Shoup encryption scheme [15] (for achieving non-malleability).

Labeled Public-Key Encryption. Labeled encryption [32] is a variation of the usual encryption notion that takes into account the presence of labels in the encryption and decryption algorithms. More precisely, both the encryption and decryption algorithms have an additional input parameter, referred to as a label, and the decryption algorithm should only correctly decrypt a ciphertext if its input label matches the label used to create that ciphertext.

The security notion for labeled encryption is similar to that of standard encryption schemes. The main difference is that, whenever the adversary wishes to ask a query to its Left-or-Right encryption oracle in the indistinguishability security game (IND-CPA) [4,26], in addition to providing a pair of messages (m_0, m_1), it also has to provide a target label ℓ to obtain the challenge ciphertext c. When chosen-ciphertext security (IND-CCA) is concerned, the adversary is also allowed to query its decryption oracle on any pair (ℓ', c') as long as $\ell' \neq \ell$ or the ciphertext c' does not match the output c of a query to its Left-or-Right encryption oracle whose input includes the label ℓ. For formal security definitions for labeled encryption schemes, please refer to [1,13].

One of the advantages of using labeled encryption, which we exploit in this paper, is that we can easily combine several IND-CCA labeled encryption schemes with the help of a strongly unforgeable one-time signature scheme so that the resulting scheme remains IND-CCA [20].

ElGamal and Cramer-Shoup Encryption. We denote by G a cyclic group of prime order q where q is large (n bits), and g a generator for this group. Let $\mathsf{pk} = (g_1, g_2, c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z, H)$ be the public key of the Cramer-Shoup scheme, where g_1 and g_2 are random group elements, x_1, x_2, y_1, y_2 and z are random scalars in $\mathbb{Z}_q$, and H is a collision-resistant hash function (actually, second-preimage resistance is enough), and $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ the associated private key. Note that (g_1, h) will also be seen as the public key of the ElGamal encryption, with z the associated private key. For the sake of simplicity, we assume in the following that public keys will additionally contain all the global parameters, such as the group G.

If $M \in G$, the multiplicative ElGamal encryption is defined as $\mathsf{EG}^{\times}_{\mathsf{pk}}(M; r) = (u_1 = g_1^r, e = h^r M)$, which can be decrypted by $M = e/u_1^z$. If $M \in \mathbb{Z}_q$, the additive ElGamal encryption is defined as $\mathsf{EG}^{+}_{\mathsf{pk}}(M; r) = (u_1 = g_1^r, e = h^r g^M)$. Note that $\mathsf{EG}^{\times}_{\mathsf{pk}}(g^M; r) = \mathsf{EG}^{+}_{\mathsf{pk}}(M; r)$. It can be decrypted after an additional discrete logarithm computation: M must be small enough. Similarly, if $M \in G$, the multiplicative labeled Cramer-Shoup encryption is defined as $\mathsf{CS}^{\times\,\ell}_{\mathsf{pk}}(M; r) = (u_1, u_2, e, v)$, such that $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r M$, $\theta = H(\ell, u_1, u_2, e)$ and $v = (cd^\theta)^r$. Decryption works as above, with $M = e/u_1^z$, but only if the ciphertext is valid: $v = u_1^{x_1 + \theta y_1} u_2^{x_2 + \theta y_2}$. If $M \in \mathbb{Z}_q$, its additive encryption $\mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M; r)$ is such that $e = h^r g^M$. The following relation holds $\mathsf{CS}^{\times\,\ell}_{\mathsf{pk}}(g^M; r) = \mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M; r)$. The decryption applies as above if M is small enough.

As already noted, from any Cramer-Shoup ciphertext (u_1, u_2, e, v) of a message M with randomness r, whatever the label ℓ is, one can extract (u_1, e) as an ElGamal ciphertext of the same message M with the same randomness r.

This extraction applies independently of the additive or multiplicative version since the decryption works the same for the ElGamal and the Cramer-Shoup ciphertexts, except for the validity check that provides the CCA security level to the Cramer-Shoup encryption scheme, whereas the ElGamal encryption scheme achieves IND-CPA security level only.

Commitments. With a commitment scheme, a player can commit to a secret value x by publishing a commitment $C = \mathsf{com}(x; r)$ with randomness r, in such a way that C reveals nothing about the secret x, which is called the *hiding* property. The player can later open C to reveal x, by publishing x and a decommitment, also referred to as witness, in a publicly verifiable way: the player cannot open C to any other value than x, which is the *binding* property. In many cases, the decommitment consists of the random r itself or some part of it. In this paper, we only consider commitment schemes in the common reference string (CRS) model in which the common parameters, referred to as the CRS, are generated honestly and available to all parties.

Note that an IND-CPA public-key encryption scheme provides such a commitment scheme: the binding property is guaranteed by the uniqueness of the plaintext (perfectly binding), and the hiding property is guaranteed by the IND-CPA security (computationally hiding). In this case, the CRS simply consists of the public-key of the encryption scheme. The Pedersen commitment $C = \mathsf{comPed}(x; r) = g^x h^r$ provides a perfectly hiding, but computationally binding commitment under the intractability of the discrete logarithm of h in basis g.

We now present additional properties that can be satisfied by the commitment. First, we say that a commitment is *extractable* if there exists an efficient algorithm, called an extractor, capable of generating a new set of common parameters (*i.e.*, a new CRS) whose distribution is equivalent to that of an honestly generated CRS and such that it can extract the committed value x from any commitment C. This is of course only possible for computationally hiding commitments, such as encryption schemes: the decryption key is the extraction trapdoor. Second, we say that a commitment is *equivocable* if there exists an efficient algorithm, called an equivocator, capable of generating a new CRS and a commitment with similar distributions to those of the actual scheme and such that the commitment can be opened in different ways. Again, this is possible for computationally binding commitments only, such as the Pedersen commitment: the knowledge of the discrete logarithm of h in basis g is a trapdoor that allows the opening of a commitment in more than one way. Finally, a *non-malleable* commitment ensures that if an adversary that receives a commitment C of some unknown value x can generate a valid commitment for a related value y, then a simulator could perform as well without seeing C. A public-key encryption scheme that is IND-CCA provides such a non-malleable commitment [22]. For formal security definitions for commitment schemes, please refer to [22,19,12].

In the following, we use encryption schemes in order to construct commitments, which immediately implies the hiding, binding and extractable properties, as said above. However, when one uses the additive versions of ElGamal or Cramer-Shoup encryption schemes, extractability (or decryption) is only possible if the committed values (or plaintexts) are small enough, hence our notion of L-extractable commitments (see Section 4) which will mean that the commitment is extractable if the committed value lies in the language L. More precisely, we will split the value to be committed in small pieces (that lie in the language L), but we will then need to be sure that they actually lie in this language to guarantee extractability. We thus introduce smooth hash functions in order to allow communications if the commitments are valid only.

3 Smooth Hash Functions on Conjunctions and Disjunctions of Languages

Smooth Projective Hash Functions. Projective hash function families were first introduced by Cramer and Shoup [16] as a means to design chosen-ciphertext secure encryption schemes. We here use the definitions of Gennaro and Lindell [22], who later showed how to use such families to build secure password-based authenticated key exchange protocols, together with non-malleable commitments. In addition to commitment schemes, we also consider here families of SPHF associated to labeled encryption as done by Canetti *et al.* [13] and by Abdalla and Pointcheval [1].

Let X be the domain of these functions and let L be a certain subset of points of this domain (a language). A key property of these functions is that, for points in L, their values can be computed by using either a *secret* hashing key or a *public* projected key. While the computation using the *secret* hashing key works for all points in the domain X of the hash function, the computation using a *public* projected key only works for points $x \in L$ and requires the knowledge of the witness w to the fact that $x \in L$. A projective hash function family is said to be *smooth* if the value of the function on inputs that are outside the particular subset L of the domain are independent of the projected key. Another important property of these functions is that, given the projected key hp, their output is uniquely defined for points $x \in L$. Moreover, if L is a *hard partitioned subset* of X (*i.e.*, it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$), this output is also *pseudo-random* if one does not know a witness w to the fact that $x \in L$ [22]. The interested reader is referred to the full version for more formal definitions.

In the particular case of the Gennaro-Lindell scheme [22], the subset $L_{\mathsf{pk},m}$ was defined as the set of $\{(c)\}$ such that c is a commitment of m using public parameters pk: there exists r for which $c = \mathsf{com}_{\mathsf{pk}}(m;r)$ where com is the committing algorithm of the commitment scheme. In the case of the CHKLM scheme [13], the subset $L_{\mathsf{pk},(\ell,m)}$ was defined as the set of $\{(c)\}$ such that c is an encryption of m with label ℓ, under the public key pk: there exists r for which $c = \mathcal{E}^{\ell}_{pk}(m;r)$ where $\mathcal{E}$ is the encryption algorithm of the labeled encryption scheme. In the case of a standard encryption scheme, the label is simply omitted. The interested reader is referred to [22,13,1] for more details.

LANGUAGES. Since we want to use more general languages, we need more detailed notations. Let **LPKE** be a labeled encryption scheme with public key pk.

Let X be the range of the encryption algorithm. Here are three useful examples of languages L in X:

– the valid ciphertexts c of m under pk, $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m)} = \{c | \exists r\ c = \mathcal{E}^{\ell}_{\mathsf{pk}}(m;r)\}$;
– the valid ciphertexts c of m_1 or m_2 under pk (that is, a disjunction of two versions of the former languages), $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_1 \vee m_2)} = L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_1)} \cup L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_2)}$;
– the valid ciphertexts c under pk, $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,*)} = \{c | \exists m\ \exists r\ c = \mathcal{E}^{\ell}_{\mathsf{pk}}(m;r)\}$.

If the encryption scheme is IND-CPA, the first two are hard partitioned subsets of X. The last one can also be a hard partitioned subset in some cases: for the Cramer-Shoup encryption, $L \subsetneq X = G^4$ and, in order to distinguish a valid ciphertext from an invalid one, one has to break the DDH problem. However, for the ElGamal encryption scheme, all the ciphertexts are valid, hence $L = X = G^2$.

More complex languages can be defined, with disjunctions as above, or conjunctions: the pairs of ciphertexts (a,b) such that $a \in L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,0\vee 1)}$ and $b \in L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,2\vee 3)}$. This set can be obtained by $(L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,0\vee 1))} \times X) \cap (X \times L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,2\vee 3)})$.

Likewise, we can define more general languages based on other primitives such as commitment schemes. The definition would be similar to the one above, with pk playing the role of the common parameters, $\mathcal{E}_{\mathsf{pk}}$ playing the role of the committing algorithm, (m,ℓ) playing the role of the input message, and c playing the role of the commitment.

More generally, in the following, we denote the language by the generic notation $L_{(\mathbf{Sch},\rho),aux}$ where aux denotes all the parameters useful to characterize the language (such as the label used, or a plaintext), ρ denotes the public parameters such as a public key pk, and **Sch** denotes the primitive used to define the language, such as an encryption scheme **LPKE** or a commitment scheme **Com**. When there is no ambiguity, the associated primitive **Sch** will be omitted.

We now present new constructions of SPHF to deal with more complex languages, such as disjunctions and conjunctions of any languages. The constructions are presented for two languages but can be easily extended to any polynomial number of languages. We then discuss about possible information leakage at the end of this section. The properties of correctness, smoothness and pseudo-randomness are easily verified by these new smooth hash systems. Due to the lack of space, the formal proofs can be found in the full version.

Conjunction of two Generic Smooth Hashes. Let us consider an encryption or commitment scheme defined by public parameters and a public key aggregated in ρ. X is the range of the elements we want to study (ciphertexts, tuples of ciphertexts, commitments, etc), and $L_1 = L_{1,\rho,aux}$ and $L_2 = L_{2,\rho,aux}$ are hard partitioned subsets of X, which specify the expected properties (valid ciphertexts, ciphertexts of a specific plaintext, etc). We consider situations where X possesses a group structure, which is the case if we consider ciphertexts or tuples of ciphertexts from an homomorphic encryption scheme. We thus denote by $\oplus$ the commutative law of the group (and by $\ominus$ the opposite operation, such that $c \oplus a \ominus a = c$).

We assume to be given two smooth hash systems SHS_1 and SHS_2, on the sets corresponding to the languages L_1 and L_2: $\mathsf{SHS}_i = \{\mathsf{HashKG}_i, \mathsf{ProjKG}_i, \mathsf{Hash}_i, \mathsf{ProjHash}_i\}$. Here, HashKG_i and ProjKG_i denote the hashing key and the projected key generators, and Hash_i and $\mathsf{ProjHash}_i$ the algorithms that compute the hash function using hk_i and hp_i respectively.

Let c be an element of X, and r_1 and r_2 two elements chosen at random. We denote by $\mathsf{hk}_1 = \mathsf{HashKG}_1(\rho, aux, r_1)$, $\mathsf{hk}_2 = \mathsf{HashKG}_2(\rho, aux, r_2)$, $\mathsf{hp}_1 = \mathsf{ProjKG}_1(\mathsf{hk}_1; \rho, aux, c)$, and $\mathsf{hp}_2 = \mathsf{ProjKG}_2(\mathsf{hk}_2; \rho, aux, c)$ the keys. A smooth hash system for the language $L = L_1 \cap L_2$ is then defined as follows, if $c \in L_1 \cap L_2$ and w_i is a witness that $c \in L_i$, for $i = 1, 2$:

$$\begin{aligned}
\mathsf{HashKG}_L(\rho, aux, r = r_1 \| r_2) &= \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)\\
\mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)\\
\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c)\\
\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; (w_1, w_2)) &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, c; w_1)\\
&\quad \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, c; w_2)
\end{aligned}$$

Disjunction of two Generic Smooth Hashes. Let L_1 and L_2 be two languages as described above. We assume to be given two smooth hash systems SHS_1 and SHS_2 with respect to these languages. We define $L = L_1 \cup L_2$ and construct a smooth projective hash function for this language as follows:

$$\begin{array}{rcll}
\mathsf{HashKG}_L(\rho, aux, r = r_1 \| r_2) & = & \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) & \\
\mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c) & = & \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) & \\
 & & \qquad \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c)) & \\
\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, c) & = & \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) & \\
\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; w) & = & \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, c; w) & \text{if } c \in L_1 \\
 & \text{or} & \mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, c; w) & \text{if } c \in L_2
\end{array}$$

where w is a witness of $c \in L_i$ for $i \in \{1, 2\}$. Then $\mathsf{ProjHash}_i(\mathsf{hp}_i; \rho, aux, c; w) = \mathsf{Hash}_i(\mathsf{hk}_i; \rho, aux, c)$. The player in charge of computing this value is supposed to know w, and in particular the language which c belongs to (the index i).

Uniformity and Independence. In the above definition of SPHF (contrarily to the original Cramer-Shoup [16] definition), the value of the projected key formally depends on the ciphertext/commitment c. However, in some cases, one may not want to reveal any information about this dependency. In fact, in certain cases such as in the construction of a SPHF for equivocable and extractable commitments in Section 5, one may not even want to leak any information about the auxiliary elements aux. When no information is revealed about aux, it means that the details about the exact language will be concealed.

We thus add a notion similar to the smoothness, but for the projected key: the projected key may or may not depend on c (and aux), but its distribution does not: Let us denote by $D_{\rho,aux,c}$ the distribution $\{\mathsf{hp} \mid \mathsf{hk} = \mathsf{HashKG}_L(\rho, aux, r)$ and $\mathsf{hp} = \mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c)\}$, on the projected keys. If, for any $c, c' \in X$, $D_{\rho,aux,c'}$ and $D_{\rho,aux,c}$ are indistinguishable, then we say that the smooth hash system has the *1-uniformity* property. If, for any $c, c' \in X$, and any auxiliary elements aux, aux', $D_{\rho,aux',c'}$ and $D_{\rho,aux,c}$ are indistinguishable, we name it *2-uniformity* property.

More than indistinguishability of distributions, the actual projected key hp may not depend at all on c, as in the Cramer and Shoup's definition. Then, we say that the smooth hash system guarantees *1-independence* (resp. *2-independence* if it does not depend on aux either). Note that the latter independence notions immediately imply the respective uniformity notions.

As an example, the smooth hash system associated with the ElGamal cryptosystem (see Section 4 page 680) guarantees 2-independence. On the other hand, the analogous system associated with the Cramer-Shoup encryption (see the full version) guarantees 2-uniformity only. For smooth hash systems combinations, one can note that in the case of disjunctions, one can get, at best, the uniformity property, since hash computations on the commitment are needed for generating the projected key. Furthermore, this is satisfied under the condition that the two underlying smooth hash systems already satisfy this property (see the full version for more details and proofs).

Finally, one should note that, in the case of disjunction, the view of the projected hash value could leak some information about the sub-language in which the input lies, if an adversary sends a fake hp_Δ. The adversary could indeed check whether $\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; w)$ equals $\mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c)$ or $\mathsf{hp}_\Delta \ominus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c)$. But first, it does not contradict any security notion for smooth hash systems; second, in all the applications below, the projected hash value is never revealed; and third, in the extractable commitments below, because of the global conjunction of the languages, an exponential exhaustive search would be needed to exploit this information, even if the committed value is a low-entropy one.

4 A Conditionally Extractable Commitment

ElGamal Commitment and Associated Smooth Hash. The ElGamal commitment is realized in the common reference string model, where the CRS ρ contains (G, pk), as defined in Section 2, for the ElGamal encryption scheme. In practice, sk should not be known by anybody, but in the security analysis, sk will be the extraction trapdoor. Let the input of the committing algorithm be a scalar $M \in \mathbb{Z}_q$. The commitment algorithm consists of choosing a random r and computing the following ElGamal encryption under random r: $C = \mathsf{EG}^+_{\mathsf{pk}}(M, r) = (u_1 = g_1^r, e = h^r g^M)$.

The smooth projective hashing, associated with this commitment scheme and the language $L = L_{(\mathbf{EG}^+,\rho),M} \subset X = G^2$ of the additive ElGamal ciphertexts C of M under the global parameters and public key defined by ρ, is the family based on the underlying ElGamal encryption scheme, as defined in [22]:

$$\begin{aligned}
\mathsf{HashKG}((\mathbf{EG}^+, \rho), M) &= \mathsf{hk} = (\gamma_1, \gamma_3) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q \\
\mathsf{Hash}(\mathsf{hk}; (\mathbf{EG}^+, \rho), M, C) &= (u_1)^{\gamma_1} (eg^{-M})^{\gamma_3} \\
\mathsf{ProjKG}(\mathsf{hk}; (\mathbf{EG}^+, \rho), M, C) &= \mathsf{hp} = (g_1)^{\gamma_1} (h)^{\gamma_3} \\
\mathsf{ProjHash}(\mathsf{hp}; (\mathbf{EG}^+, \rho), M, C; r) &= (\mathsf{hp})^r
\end{aligned}$$

First, under the DDH problem (semantic security of the ElGamal encryption scheme), L is a hard partitioned subset of $X = G^2$. Then, for $C = \mathsf{EG}^+_{\mathsf{pk}}(M, r)$, and thus with the witness r, the algorithms are defined as above using the same notations as in [22].

L-extractable Commitments. Note that the value g^M would be easily extractable from this commitment (seen as the multiplicative ElGamal encryption). However, one can extract M itself (the actual committed value) only if its size is small enough so that it can be found as a solution to the discrete logarithm problem. In order to obtain "extractability" (up to a certain point, see below), one should rather commit to it in a bit-by-bit way.

Let us denote $M \in \mathbb{Z}_q$ by $\sum_{i=1}^{m} M_i \cdot 2^{i-1}$, where $m \leq n$. Its commitment is $\mathsf{comEG}_{\mathsf{pk}}(M) = (b_1, \ldots, b_m)$, where $b_i = \mathsf{EG}^+_{\mathsf{pk}}(M_i \cdot 2^{i-1}, r_i) = (u_{1,i} = {g_1}^{r_i}, e_i = h^{r_i} g^{M_i \cdot 2^{i-1}})$, for $i = 1, \ldots, m$. The homomorphic property of the encryption scheme allows to obtain, from this tuple, the above simple commitment of M

$$C = \mathsf{EG}^+_{\mathsf{pk}}(M, r) = (u_1, e) = (\prod u_{1,i}, \prod e_i) = \prod b_i, \text{ for } r = \sum r_i.$$

We now precise what we mean by "extractability": Here, the commitment will be extractable if the messages M_i are bits (or at least small enough), but we cannot ensure that it will be extractable otherwise. More generally, this leads to a new notion of $L-$*extractable commitments*, which means that we allow the primitive not to be extractable if the message does not belong to a certain language L (*e.g.* the language of encryptions of 0 or 1), which is informally the language of all commitments valid and "of good shape", and is included into the set X of all commitments.

Smooth Hash Functions. For the above protocol, we need a smooth hash system on the language $L = L_1 \cap L_2$, where $L_1 = \{(b_1, \ldots, b_m) \mid \forall i, b_i \in L_{(\mathbf{EG}^+,\rho),0\vee 1}\}$, $L_2 = \{(b_1, \ldots, b_m) \mid C = \prod_i b_i \in L_{(\mathbf{EG}^\times,\rho),g^M}\}$, to within a factor (corresponding to the offest 2^{i-1}) with

$$\begin{array}{ll} L_{(\mathbf{EG}^+,\rho),0\vee 1} = L_{(\mathbf{EG}^+,\rho),0} \cup L_{(\mathbf{EG}^+,\rho),1} & L_{(\mathbf{EG}^+,\rho),0} = \{C \mid \exists r\ C = \mathsf{EG}^+_{\mathsf{pk}}(0, r)\} \\ L_{(\mathbf{EG}^\times,\rho),g^M} = \{C \mid \exists r\ C = \mathsf{EG}^\times_{\mathsf{pk}}(g^M, r)\} & L_{(\mathbf{EG}^+,\rho),1} = \{C \mid \exists r\ C = \mathsf{EG}^+_{\mathsf{pk}}(1, r)\} \end{array}$$

It is easy to see that this boils down to constructing a smooth hash system corresponding to a conjunction and disjunction of languages, as presented in the previous section.

Certification of Public Keys

Description. A classical application of extractable commitments is in the certification of public keys (when we want to be sure that a person joining the system actually knows the associated private key). Suppose that a user U owns a pair of secret and public keys, and would like to have the public key certified by the authority. A natural property is that the authority will not certify this public key unless it is sure that the user really owns the related private key, which is usually ensured by a zero-knowledge proof of knowledge: the user knows the private key if a successful extractor exists.

Here we present a construction that possesses the same property without requiring any explicit proof of knowledge, furthermore in a concurrent way since there is no need of any rewinding:

- First, the user sends his public key g^M, along with a bit-by-bit L-extractable commitment of the private key M, i.e. a tuple $\mathsf{comEG}_{\mathsf{pk}}(M) = (b_1, \dots, b_m)$ as described above, from which one can derive $C = \prod b_i = \mathsf{EG}^+_{\mathsf{pk}}(M, r) = \mathsf{EG}^\times_{\mathsf{pk}}(g^M, r)$.
- We define the smooth hash system related to the language $L_1 \cap L_2$, where $L_1 = \cap_i L_{1,i}$, with $L_{1,i}$ the language of the tuples where the i-th component b_i is an encryption of 0 or 1, and L_2 is the language of the tuples where the derived $C = \prod b_i$ is an encryption of the public key g^M (under the multiplicative ElGamal, as in Section 4 page 680).

  Note that when the tuple $(b_1, \dots, b_m)$ lies in $L_1 \cap L_2$, it really corresponds to an extractable commitment of the private key M associated to the public key g^M: each b_i encrypts a bit, and can thus be decrypted, which provides the i-th bit of M.
- The authority computes a hash key hk, the corresponding projected key hp on $(b_1, \dots, b_m)$ and the related hash value Hash on $(b_1, \dots, b_m)$. It sends hp to U along with $\mathsf{Cert} \oplus \mathsf{Hash}$, where Cert is the expected certificate. Note that if Hash is not large enough, a pseudo-random generator can be used to expand it.
- The user is then able to recover his certificate if and only if he can compute Hash: this value can be computed with the algorithm $\mathsf{ProjHash}$ on $(b_1, \dots, b_m)$, from hp. But it also requires a witness w proving that the tuple $(b_1, \dots, b_m)$ lies in $L_1 \cap L_2$.

With the properties of the smooth hash system, if the user correctly computed the commitment, he knows the witness w, and can get the same mask Hash to extract the certificate. If the user cheated, the smoothness property makes Hash perfectly unpredictable: no information is leaked about the certificate.

Security Analysis. Let us outline the security proof of the above protocol. First, the security model is the following: no one can obtain a certificate on a public key if it does not know the associated private key (that is, if no simulator can extract the private key). In other words, the adversary wins if it is able to output (g^M, Cert) and no simulator can produce M.

The formal attack game can thus be described as follows: the adversary $\mathcal{A}$ interacts several times with the authority, by sending public keys and commitments, and asks for the corresponding certificates. It then outputs a pair (g^M, Cert) and wins if no simulator is able to extract M from the transcript.

The simulator works as follows: it is given access to a certification (signing) oracle, and generates a pair of public and private keys $(\mathsf{sk}, \mathsf{pk})$ for the ElGamal encryption. The public key is set as the CRS that defines the commitment scheme. The private key will thus be the extraction trapdoor.

When the simulator receives a certification request, with a public key and a commitment, it first tries to extract the associated private key, granted the extraction trapdoor. In case of success, the simulator asks the signing oracle to provide it with the corresponding certificate on the public key, and complete the process as described in the protocol. However, extraction may fail if the commitments are not well constructed (not in $L_1 \cap L_2$). In such a case, the simulator sends back a random bit-string of appropriate length. In case of successful extraction, the answer received by the user is exactly the expected one. In case of failure, it is perfectly indistinguishable too since the smoothness property of the hash function would make a perfectly random mask Hash (since the input is not in the language).

After several interactions, $\mathcal{A}$ outputs a pair (g^M, Cert), which is forwarded by the simulator. Either g^M has been queried to the signing oracle, which means that the extraction had succeeded, the simulator knows M and the adversary did not win the attack game, or this is a valid signature on a new message: existential forgery under chosen-message attack.

5 A Conditionally Extractable Equivocable Commitment

In this section, we enhance the previous commitment schemes with equivocability, which is not a trivial task when one wants to keep the extraction property. Note that we first build a malleable extractable and equivocable commitment using the ElGamal-based commitment (see Section 4 page 680), but one can address the non-malleability property by simply building the commitment upon the Cramer-Shoup encryption scheme. All the details of this extension are given in the full version. In the following, if b is a bit, we denote its complement by $\overline{b}$ (i.e., $\overline{b} = 1 - b$). We furthermore denote by $x[i]$ the i^{th} bit of the bit-string x.

Equivocability. Commitments that are both extractable and equivocable seem to be very difficult to obtain. Canetti and Fischlin [12] proposed a solution but for one bit only. Damgård and Nielsen [17] proposed later another construction. But for efficiency reasons, in our specific context, we extend the former proposal. In this section, we thus enhance our previous commitment (that is already L-extractable) to make it equivocable, using the Canetti and Fischlin's approach. Section 5 page 686 will then apply a non-malleable variant of our new commitment together with the associated smooth hash function family in order to build a password-authenticated key exchange protocol with adaptive security in the UC framework [11]. The resulting protocol is reasonably efficient and, in particular, more efficient than the protocol by Barak *et al.* [3], which to our knowledge is the only one achieving the same level of security in the standard model.

Description of the Commitment. Our commitment scheme is a natural extension of Canetti-Fischlin commitment scheme [12], in a bit-by-bit way. It indeed uses the ElGamal public-key encryption scheme, for each bit of the bit-string. Let $(y_1, \ldots, y_m)$ be random elements in G. This commitment is realized in the common reference string model, the CRS ρ contains (G, pk), where pk is an ElGamal public key and the private key is unknown to anybody, except to the commitment extractor. It also includes this tuple $(y_1, \ldots, y_m)$, for which the discrete logarithms in basis g are unknown to anybody, except to the commitment equivocator. Let the input of the committing algorithm be a bit-string $\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}$. The algorithm works as follows:

– For $i = 1, \dots, m$, it chooses a random value $x_{i,\pi_i} = \sum_{j=1}^{n} x_{i,\pi_i}[j] \cdot 2^{j-1}$ and sets $x_{i,\overline{\pi}_i} = 0$.
– For $i = 1, \dots, m$, the algorithm commits to π_i, using the random x_{i,π_i}: $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}) = g^{x_{i,\pi_i}} y_i^{\pi_i}$ and defining $\mathbf{a} = (a_1, \dots, a_m)$.
– For $i = 1, \dots, m$, it computes the ElGamal commitments (see the previous section) of $x_{i,\delta}$, for $\delta = 0, 1$: $(\mathbf{b}_{i,\delta} = (b_{i,\delta}[j])_j = \mathsf{comEG}_{\mathsf{pk}}(x_{i,\delta})$, where $b_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j])$. One can directly extract from the computation of the $b_{i,\delta}[j]$ an encryption $B_{i,\delta}$ of $x_{i,\delta}$: $B_{i,\delta} = \prod_j b_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}, r_{i,\delta})$, where $r_{i,\delta}$ is the sum of the random coins $r_{i,\delta}[j]$.

The entire random string for this commitment is (where n is the bit-length of the prime order q of the group G) $R = (x_{1,\pi_1}, (r_{1,0}[1], r_{1,1}[1], \dots, r_{1,0}[n], r_{1,1}[n]), \dots, x_{m,\pi_m}, (r_{m,0}[1], \dots, r_{m,1}[n]))$. From which, all the values $r_{i,\overline{\pi}_i}[j]$ can be erased, letting the opening data (witness of the committed value) become limited to $\mathsf{w} = (x_{1,\pi_1}, (r_{1,\pi_1}[1], \dots, r_{1,\pi_1}[n]), \dots, x_{m,\pi_m}, (r_{m,\pi_m}[1], \dots, r_{m,\pi_m}[n]))$. The output of the committing algorithm, of the bit-string π, using the random R, is $\mathsf{com}_\rho(\pi; R) = (\mathbf{a}, \mathbf{b})$, where $\mathbf{a} = (a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}))_i, \mathbf{b} = (b_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j]))_{i,\delta,j}$.

Opening. In order to open this commitment to π, the above witness w (with the value π) is indeed enough: one can build again, for all i and j, $b_{i,\pi_i}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\pi_i}[j] \cdot 2^{j-1}, r_{i,\pi_i}[j])$, and check them with $\mathbf{b}$. One can then also compute again all the $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i})$, and check them with $\mathbf{a}$. The erased random elements would help to check the encryptions of zeroes, what we do not want, since the equivocability property will exploit that.

Properties. Let us briefly check the security properties, which are formally proven in the full version. First, because of the perfectly hiding property of the Pedersen commitment, unless some information is leaked about the $x_{i,\delta}[j]$'s, no information is leaked about the π_i's. And granted the semantic security of the ElGamal encryption scheme, the former privacy is guaranteed. Since the Pedersen commitment is (computationally) binding, the a_i's cannot be opened in two ways, but only one pair (π_i, x_{i,π_i}) is possible. Let us now consider the new extended properties:

– (conditional) extractability is provided by the bit-by-bit encryption. With the decryption key sk, one can decrypt all the $b_{i,\delta}[j]$, and get the $x_{i,\delta}$ (unless the ciphertexts contain values different from 0 and 1, which will be one condition for extractability). Then, one can check, for $i = 1, \dots, m$, whether $a_i = \mathsf{comPed}(0, x_{i,0})$ or $a_i = \mathsf{comPed}(1, x_{i,1})$, which provides π_i (unless none of the equalities is satisfied, which will be another condition for extractability).
– equivocability is possible using the Pedersen commitment trapdoor. Instead of taking a random x_{i,π_i} and then $x_{i,\overline{\pi}_i} = 0$, which specifies π_i as the committed bit, one takes a random $x_{i,0}$, computes $a_i = \mathsf{comPed}(0, x_{i,0})$, but also extracts $x_{i,1}$ so that $a_i = \mathsf{comPed}(1, x_{i,1})$ too (which is possible with the knowledge of discrete logarithm of y_i in basis g, the trapdoor). The rest of the commitment procedure remains the same, but now, one can open any

bit-string for π, using the appropriate x_{i,π_i} and the corresponding random elements (the simulator did not erase).

The Associated Smooth Projective Hash Function. As noticed above, our new commitment scheme is conditionally extractable (one can recover the $x_{i,\delta}$'s, and then the committed value π), under the conditions that all the ElGamal ciphertexts encrypt either 0 or 1, and the a_i is a commitment of either 0 or 1, with random $x_{i,0}$ or $x_{i,1}$.

As before, one wants to make the two hash values (direct computation and the one from the projected key) be the same if the two parties use the same input π and perfectly independent if they use different inputs (smoothness). One furthermore wants to control that each a_i is actually a Pedersen commitment of π_i using the encrypted random x_{i,π_i}, and thus $g^{x_{i,\pi_i}} = a_i/y_i^{\pi_i}$: the extracted x_{i,π_i} is really the private key M related to a given public key g^M that is $a_i/y_i^{\pi_i}$ in our case. Using the same notations as in Section 4 page 680, we want to define a smooth hash system showing that, for all i, δ, j, $b_{i,\delta}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee 1}$ and, for all i, $B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),(a_i/y_i{}^{\pi_i})}$, where $B_{i,\pi_i} = \prod_j b_{i,\pi_i}[j]$.

Combinations of these smooth hashes. Let C be the above commitment of π using randomness R as defined in Section 5 page 683. We now precise the language $L_{\rho,\pi}$, consisting informally of all the valid commitments "of good shape":

$$L_{\rho,\pi} = \left\{ C \;\middle|\; \begin{array}{ll} \exists R \text{ s. t. } C = \mathsf{com}_\rho(\pi, R) & \text{and } \forall i\ \forall j\ \ b_{i,\pi_i}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee 1} \\ & \text{and } \forall i\ \ B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),a_i/y_i{}^{\pi_i}} \end{array} \right\}$$

The smooth hash system for this language relies on the smooth hash systems described previously, using the generic construction for conjunctions and disjunctions as described in Section 3. The precise definition of this language (which is constructed from conjunctions and disjunctions of simple languages) can be found in the full version, omitting the labels and replacing the Cramer-Shoup encryption $\mathbf{CS}^+$ by the ElGamal one $\mathbf{EG}^+$.

Properties: Uniformity and Independence. With a non-malleable variant of such a commitment and smooth hash function, it is possible to improve the establishment of a secure channel between two players, from the one presented Section 4 page 681. More precisely, two parties can agree on a common key if they both share a common (low entropy) password π. However, a more involved protocol than the one proposed in Section 4 is needed to achieve all the required properties of a password-authenticated key exchange protocol, as it will be explained in Section 5 page 686 and proven in the full version.

Nevertheless, there may seem to be a leakage of information because of the language that depends on the input π: the projected key hp seems to contain some information about π, that can be used in another execution by an adversary. Hence the independence and uniformity notions presented Section 3 page 679, which ensure that hp does not contain any information about π. Proofs of these properties can be found in the full version.

Estimation of the Complexity. Globally, each operation (commitment, projected key, hashing and projected hashing) requires $\mathcal{O}(mn)$ exponentiations in G, with small constants (at most 16).

UC-Secure PAKE with Adaptive Security. The primitive presented above, but using the Cramer-Shoup encryption scheme (as described in the full version) is a non-malleable conditionally extractable and equivocable commitment. We now sketch how to use this new primitive in order to construct the first efficient adaptively-secure password-authenticated key exchange protocol in the UC framework with erasures. For lack of space, all the details can be found in the full version. The passwords are not known at the beginning of the simulation: $\mathcal{S}$ will manage to correct the errors (thanks to the equivocability) but without erasures there would remain clues on how the computations were held, which would give indications on the passwords used.

Our protocol is based on that of Gennaro and Lindell [22]. At a high level, the players in the KOY/GL protocol exchange CCA-secure encryptions of the password, under the public-key found in the common reference string, which are essentially commitments of the password. Then, they compute the session key by combining smooth projective hashes of the two password/ciphertext pairs. The security of this protocol relies on the properties of smoothness and pseudo-randomness of the smooth projective hash function. But as noted by Canetti *et al* in [13], the KOY/GL protocol is not known to achieve UC security: the main issue is that the ideal-model simulator must be able to extract the password used by the adversary before playing, which is impossible if the simulator is the initiator (on behalf of the client), leading to such situation in which the simulator is stuck with an incorrect ciphertext and will not be able to predict the value of the session key.

To overcome this problem, the authors of [13] made the client send a pre-flow which also contains an encryption of the password. The server then sends its own encryption, and finally the client sends another encryption, as well as a zero-knowledge proof showing that both ciphertexts are consistent and encrypt the same password. This time the simulator, playing as the client or the server, is able to use the correct password, recovered from the encrypted value sent earlier by the other party. The pre-flow is never used in the remaining of the protocol, hence the simulator can send a fake one, and simulate the zero-knowledge proof.

Unfortunately, the modification above does not seem to work when dealing with adaptive adversaries, which is the case in which we are interested. This is because the simulator cannot correctly open the commitment when the adversary corrupts the client after the pre-flow has been sent. A similar remark applies to the case in which the server gets corrupted after sending its first message. As a result, in addition to being extractable, the commitment scheme also needs to be equivocable for the simulator to be able to provide a consistent view to the adversary. Since the use of the equivocable and extractable commitment schemes also seems to solve the problem of proving the original Gennaro-Lindell protocol secure in the UC model, we opted to use that protocol as the starting point of our protocol.

These remarks are indeed enough (along with minor modifications) to obtain adaptive security. Thus, our solution essentially consists in using our non-malleable extractable and equivocable commitment scheme in the Gennaro-Lindell protocol when computing the first two flows. As presented in the previous subsections, extractability may be conditional: We include this condition in the language of the smooth hash function (note that the projected keys sent do not leak any information about the password). Additional technical modifications were also needed to make things work and can be found in the full version.

Acknowlegments

This work was supported in part by the French ANR-07-SESU-008-01 PAMPA Project, and the European ECRYPT Project.

References

1. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer, Heidelberg (2006)
2. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
3. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, pp. 394–403. IEEE Computer Society Press, Los Alamitos (1997)
5. 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)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
7. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
8. Bohli, J.-M., Gonzalez Vasco, M.I., Steinwandt, R.: Password-authenticated constant-round group key establishment with a common reference string. Cryptology ePrint Archive, Report 2006/214 (2006)
9. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
10. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
11. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)

12. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
13. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
14. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
15. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
16. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
17. Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
18. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
19. Di Crescenzo, G., Katz, J., Ostrovsky, R., Smith, A.: Efficient and non-interactive non-malleable commitment. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 40–59. Springer, Heidelberg (2001)
20. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)
21. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
22. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
23. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
24. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: 19th ACM STOC, pp. 218–229. ACM Press, New York (1987)
25. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the ACM 38(3), 691–729 (1991)
26. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
27. Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer, Heidelberg (2005)

28. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
29. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
30. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: 12th SODA, January 2001, pp. 448–457. ACM-SIAM (2001)
31. 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)
32. Shoup, V.: ISO 18033-2: An emerging standard for public-key encryption. Final Committee Draft (December 2004)

Author Index