Lecture Notes in Computer Science 2357

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Volume Editor

Matt Blaze
AT&T Labs-Research
Room A275, 180 Park Ave., P.O. Box 971
Florham Park, NJ 07932-0971, USA
E-mail: mab@research.att.com

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

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

ISSN 0302-9743
ISBN 3-540-00646-X Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna e. K.
Printed on acid-free paper SPIN: 10870172 06/3142 5 4 3 2 1 0

Preface

The Sixth International Financial Cryptography Conference was held during March 11–14, 2002, in Southampton, Bermuda. As is customary at FC, these proceedings represent "final" versions of the papers presented, revised to take into account comments and discussions from the conference.

Submissions to the conference were strong, with 74 papers submitted and 19 accepted for presentation and publication. (Regrettably, three of the submitted papers had to be summarily rejected after it was discovered that they had been improperly submitted in parallel to other conferences.) The small program committee worked very hard under a tight schedule (working through Christmas day) to select the program. No program chair could ask for a better committee; my thanks to everyone for their hard work and dedication.

In addition to the refereed papers, the program included a welcome from the Minister of Telecommunications and e-Commerce, Renee Webb, a keynote address by Nigel Hickson, and a panel on privacy tradeoffs chaired by Rebecca Wright (with panelists Ian Goldberg, Ron Rivest, and Graham Wood). The traditional Tuesday evening "rump session" was skillfully officiated by Markus Jakobsson.

My job as program chair was made much, much easier by the excellent work of our general chair, Nicko van Someren, who performed the miracle of hiding from me any evidence of the innumerable logistical nightmares associated with conducting this conference. I have no idea how he did it, but it must have involved many sleepless nights.

June 1, 2002 Matt Blaze

Program Committee

Matt Blaze	(AT&T Labs)
Program Chair	
Dan Boneh	(Stanford University)
Stefan Brands	(Zero Knowledge)
Dan Geer	(@stake)
Ian Goldberg	(Zero Knowledge)
Angelos Keromytis	(Columbia University)
Paul Kocher	(Cryptography Research)
Ron Rivest	(MIT)
Tomas Sander	(Intertrust)
Rebecca Wright	(AT&T Labs)

General Chair

Nicko van Someren (nCipher) ll

Table of Contents

E-voting without 'Cryptography'
(Extended Abstract)

Dahlia Malkhi, Ofer Margo, and Elan Pavlov

School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel
{dalia,omargo,elan}@cs.huji.ac.il

Abstract. We present a new, distributed e-voting protocol, which does not require any 'conventional' cryptography, or other means of non-trivial math, and is information-theoretically secure. In our system, the voter receives a receipt confirming her vote, and does not need to trust any pollster to correctly encrypt her vote, or convey it to the talliers. The system can withstand corruption of almost half the number of authorities running the elections. To achieve these goals, we present *enhanced check vectors*, a new weak signature scheme as well as a new protocol for joint creation of weak signatures by multiple servers.

1 Introduction

The need for conducting electronic elections received reiterated emphasis in the recent U.S. presidential elections, and the wide-spread dispute it caused over the Bush-Gore results. There is a clear desire for a technology which makes it easy for every citizen to cast his/her vote and verify the election results, while eliminating the risk of fraud or ambiguous ballots [CM01].

Such electronic voting schemes are required to guarantee the privacy of voters, the verifiability of the results, to prevent ballot stuffing and to ensure that all eligible voters are able to vote. However, all computer-based voting systems rely on some piece of software, called the *pollster*, to interact with the human voter in order to accept his/her vote in the clear. The pollster then transmits the vote (encrypted, broken into pieces, etc.) to the voting system. The easiest attack on any electronic voting system is to modify the pollster software. For example, the pollster may be used for snooping to votes if it is installed on the home PC of voters. Or, it can be used to alter or omit undesired votes (simply by 'pretending' that the voter punched different keys). As pointed by various state and federal committees, [IPI01,CIV01] this limitation stands in the way of deployment of electronic voting, particularly over the Internet, as a pollster would need to be installed on the untrusted environment of home PCs.

In this paper, we present an electronic voting scheme that does not require conventional cryptography on the users' side, and hence, does not require a trusted pollster. The solution employs two novel tools: One is an extension of Rabin and Ben-Or's *check vectors* [RB89], which allows one party to hand another party a secret that is guaranteed to be accepted by a group of verifiers.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 1–15, 2003.

The other is a full fledged electronic voting protocol that makes use of enhanced check vectors and an *anonymous multiparty computation* (AMPC) [MP01] in order to provide a user with a cryptography-free receipt upon voting. The receipt proves to the voter that his/her vote is recorded correctly - no pollster (or any other component) is capable of forging such a receipt if the vote was modified. Our solution decentralizes all phases of the voting process, including registration, such that no collusion of a threshold of authorities can compromise the accuracy of the tally or the privacy of users.

In addition to the usual properties of privacy, democracy, verifiability and accuracy, our approach has the following desirable features:

- It does not use any conventional means of cryptography, polynomial secret sharing, or other non-trivial math. Instead, we rely on AMPC and a new weak signatures primitive, enhanced check vectors, to achieve our goals. The voter is not required to perform complicated computations, and needs only perform $O(b)$ multiplications and additions in order to verify that she holds a valid voting credential, or that her ballot was correctly accepted by the talliers (where b is a security parameter).
- The system can withstand up to $b = \lceil \frac{m}{2} \rceil - 1$ corrupt authorities, where m is the number of talliers or registrars. In particular, we distribute the registration process so that a coalition of b or fewer registrars are not able to compromise voters privacy or to stuff the ballot box.
- The privacy provided by our protocol is information-theoretic, not computational.
- There is no single trusted piece of software that needs to receive a voter's ballot in the clear. Hence, no piece of software can pretend that a different key was "punched" by the voter undetectably.
- A voter can verify that she holds a valid voting credential before the elections and can verify that her vote was properly accepted by the talliers during the election process (that is, a voter gets an election *receipt*). This prevents a faulty pollster from modifying or obstructing the vote.
- A pollster by himself can not tell which candidate the voter voted for (even if he is able to monitor and record all of the voter communication during election day). A pollster requires the cooperation of at least one tallier in order to know which candidate the voter voted for. [1]
- The efficiency of our scheme can be summarized as follows: When there are m tallying and m registration authorities, n voters, and for a security parameter $b \leq \frac{m}{2} - 1$, such that the system can withstand up to b corrupt authorities, the total amount of communication is $O(nb^2)$ during the election phase, and $O(nmb)$ during the registration phase. More specifically, the voter holds one ballot of length $b + 2$, which on election day she needs to send to any b different talliers. Each tallier is required to perform $O(nb)$ multiplications and additions during the election themselves. The voter may perform $O(b^2)$ multiplications and additions in order to verify that she holds a valid ballot,

[1] Our protocol could be enhanced so that a pollster cooperating with up to b talliers would not be able to know which candidate the voter voted for, by splitting each ballot among talliers. For simplicity, this is not pursued further in this paper.

or that her ballot was properly received by the talliers. (The computation required is $O(b)$ multiplications and additions per tallier). Other servers may need to perform up to $O(nmb)$ multiplications and additions during the registration stage.

Technical approach: In order to provide cryptography-free ballots and receipts, we make use of a novel information-theoretic checking protocol that extends the *check vectors* of Rabin and Ben-Or [RB89]. In their information checking protocol, there are three parties: A dealer DLR, a receiver RCV and an intermediary INT. INT obtains a secret V from DLR, and would like to give it at a later time to RCV. RCV should be able to confirm that the secret V indeed originated with DLR, in order to accept the secret. Also, INT should know with high probability whether RCV will accept the secret or not.

Our *enhanced check vectors* protocol extends the check vector mechanism in two ways. First, it accommodates a group of receivers such that each receiver can verify independently that the secret indeed originated with DLR, however, no collusion of b or fewer of the receivers can reconstruct the secret V using the information (check vectors) they hold. The intermediary uses the same secret with all receivers, and (unlike in [RB89]) does not need to maintain a separate version of the secret for each potential receiver. Second, we show how the secret can be jointly generated and distributed by multiple dealers, such that intermediaries remain anonymous and such that a collusion of b or fewer of dealers and receivers are not able to recover secrets, forge them, or relate them to intermediaries. We employ an anonymous multiparty computation (AMPC) [MP01] to accomplish the last part.

We proceed to show how the enhanced check vectors protocol can be used in a distributed on-line election protocol whose essence is as follows: Before the elections, each voter receives one vote vector V for each possible ballot value s, e.g., one for Bush and one for Gore. For each vote vector issued to a voter, a corresponding check vector B is issued to each tallier. On elections day, the voter anonymously sends the vote vector V corresponding to her desired ballot s to a minimum of $b+1$ different talliers. Each tallier verifies that $s = BV$ is in S to confirm the eligibility of the voter to vote.

Using the enhanced check vectors protocol, a voter may verify before the elections that any vector it holds is valid and will be accepted by talliers. Moreover, the voter makes use of check vectors to obtain a receipt for her vote. This is achieved as follows. For each possible vote value s, a voter is issued several pairs of vote vectors, $\{\langle V_{1,0}, V_{1,1}\rangle, \langle V_{2,0}, V_{2,1}\rangle, ...\}$. Before the elections, the voter sends one of the vectors, e.g., $V_{1,0}$ to all the talliers, and expects to receive in return the check vector $B_{1,1}$ corresponding to the second vote vector in the pair, $V_{1,1}$. She then verifies that $V_{1,1}B_{1,1} = s$. Receipt delivery during the elections is done in the same manner.

We note that while none of the elements in the protocol requires the use of cryptography, we work in the classic *secure channels* model that requires that the different authorities participating in the protocol should be able to communicate with one another and with the user in a secure manner. In practice, cryptography

may be deployed in order to secure the links, or they may be secured by benign means (e.g., regular mail, telephone).

The rest of the paper is organized as follows: In section 1.1 we discuss Related Work. In section 2 we describe the enhanced check vectors protocol and the mechanism used to jointly generate and distribute the secret vote vectors and check vectors. In section 3 we describe the actual voting protocol. We then follow with a security analysis of our work (Section 4) and some concluding remarks (Section 5).

1.1 Related Work

In most general terms, electronic voting protocols can be divided into two categories, centralized and decentralized.

Centralized voting schemes rely on the existence of an anonymous communication channel between voters and either the registration or the tallying authorities in order to maintain voters privacy. The first to propose a method utilizing anonymous communication for electronic voting was Chaum, who has suggested in 1985 [Cha85] the use of *blind signatures* as a tool for privacy in electronic cash and electronic voting systems. The first actual voting protocol employing blind signatures appeared in 1992 [FOO92] and the first published implementation in 1997 [CC97]. A somewhat improved version was offered by He & Su in 98 [HS98]. The basic idea underlying all of these schemes is to employ two logical authorities, a registrar and a tallier. The registrar validates eligible voters and provides them with anonymized certified voting tags, using blind signatures. The voters then cast their (blindly signed) ballots over anonymous communications channels to the tallier. Some of the problems exhibited by these methods that our approach addresses are:

- The voters must fully trust a *pollster* - a piece of software that carries out the (non trivial) encryption functions and anonymous communications for them.
- The registrar, or the registrar in collusion with the tallier, can stuff the ballot box with invalid votes. We note that the idea of using multiple registrars to address this problem has been suggested before by DuRette in [DuR99].
- They require the use of anonymous channels which are costly to maintain, and also exhibit various difficulties in reality (see [WALS02] for a good discussion of the degradation in anonymity protection of various anonymizers).

The second category of electronic voting schemes, decentralized methods, includes various secret-sharing based schemes, e.g., [CFSY96,CGS97,Sch99] (most of them based on the extrapolations of polynomials). In such schemes, a collection of authorities are employed to reduce the trust placed in any single (tallier) authority. The paradigm works as follows: At the voting stage, voters split their ballots into shares using secret sharing techniques, and secretly send each such share to a single authority. At the tallying stage, each authority separately combines the shares of all ballots into a share of the result. Using the homomorphic properties of the secret sharing techniques, the authorities then jointly compute

the results of the elections, such that a collusion of fewer than a threshold of the tallying authorities does not reveal information about any single ballot. As these schemes do not rely on anonymous channels, they are quite practical and are in fact deployed in a commercial system, e.g. `VoteHere.Net`. Two of the above problems remain here as well:

- Voters need to trust a pollster that will carry out the necessary encryption and secret sharing for them.
- The issue of registration is generally left outside the scope of such systems, and thus the problem of the registration authority stuffing votes remains.

A different kind of voting method that is not included in either of the categories above is the decentralized protocol in [DmLM82]. This method works without involving any external authority, and moreover, prevents stuffing, but it assumes that voters can know and verify each other and 100% voter participation. Furthermore, it requires a number of rounds of communications and computations among all participants, which is linear in the number of participants. It is therefore not feasible to deploy it in a realistic, large scale elections.

The information checking protocol and check vectors that our approach is based on were first introduced by Rabin and Ben-Or in [RB89] in the context of verifiable secret sharing and secure multiparty computation. Their basic version of check vectors supports only one receiver, who is able to verify the secret as follows. The DLR hands out a pair (s, y) to INT, and a pair (b, c) to RCV (the check vector). RCV verifies when receiving the secret that $s + by = c$. In this (basic) method, multiple receivers using different check vectors are able to recover the original secret by linear extrapolation. Rabin & Ben-Or introduce an extension which supports multiple receivers but requires INT to use a different random value, y_i, for each potential receiver (each receiver verifies $s + b_i y_i = c_i$). Translated to our setting, this would imply that a voter holds a different ballot for each potential tallier. In contrast, in our scheme the voter holds only one ballot, whose length is determined solely by the security parameter.

Franklin and Yung [FY94] treat check vectors as a form of *weak signatures* (signatures that can be verified only by certain checking centers) and show how the protocol can be extended to support blind weak signatures. They proceed to show how these weak signatures can be used for digital cash. They also mention the possibility of using multiple receivers (*checking centers*), using Rabin & Ben-Or's extension to the check vector protocol, and using a majority agreement between the checking centers, in order to decide whether the input (the secret) is valid or not.

AMPC is a new approach for Anonymity without Cryptography, suggested by Malkhi & Pavlov [MP01]. In an AMPC, there are several levels (rows) of players, each row basically serving as the analog of one MixNet server [Cha81]. To use AMPC, users split their input into several shares, and submit each share to one of the players in the the first (top) row. The top row players permute the input shares they receive (using an agreed-upon permutation). Then each player splits his share into several sub-shares, and passes each sub-share to a player in the second level. Each player in the second level combines the sub-shares it received

into one share, permutes its shares and again passes split shares to the players in the third level. The process continues until all players in the last (bottom) level send their shares to the receiver, which combines the shares to obtain the original input values. Malkhi and Pavlov briefly discuss in [MP01] an election protocol, which can be used with AMPC. However, their protocol does not enable voters to verify that talliers properly received their ballots during the elections, neither does it enable voters to verify that they hold valid election credentials before the elections. Thus, their scheme does not fully support the requirement of democracy (guaranteeing that each voter is able to vote). Also, their protocol is relatively complicated compared to ours, and requires the talliers and registrars to perform commitments on the voter credentials and ballots before they are computed and published.

2 The Enhanced Check Vectors Protocol

In the enhanced check vectors protocol we have a dealer DLR, who would like to give a secret V with a meaning s to intermediary INT, where s is chosen from a small set of possible meanings S (relative to the size of Z_q). At a later time, INT would like to forward the secret to one or more receivers, RCV_1, ..., RCV_m. Each one of the receivers should be able to independently verify the secret meaning and the fact that it indeed originated with DLR. However, a subset of b or fewer receivers (where b is a security parameter of the system) should not be able to reconstruct the secret V. In addition, INT should be able to verify, with high probability, that the receivers would accept her secret.

The basic protocol proceeds as follows (all additions and multiplications are done in Z_q):

- To hand over a secret V with a meaning s, the dealer generates a random vector V of minimal length $b+1$ and gives it to INT. It also generates a group of check vectors $\{B_1..B_m\}$, one vector for each potential receiver. The check vectors are generated so that $VB_i = s$ (scalar product) for all $1 \leq i \leq m$,
- To give V to a receiver, INT sends V to the receiver. Assuming that the receiver holds a check vector B for this secret, it computes $s = BV$ and verifies that s is in S.

In order to enable INT to verify that her secret would be accepted by RCV, we do the following: For each meaning s, the dealer issues several pairs of vectors as described above, say $\{\langle V_{1,0}, V_{1,1}\rangle, \langle V_{2,0}, V_{2,1}\rangle, ...\}$ to INT, and gives each receiver corresponding pairs of check vectors, $\{\langle B_{1,0}, B_{1,1}\rangle, \langle B_{2,0}, B_{2,1}\rangle, ...\}$. All vectors and corresponding check vectors satisfy $V_{k,j}B_{k,j} = s$. To confirm that it holds valid vectors with high probability, INT chooses some $V_{k,j}$ at random, sends it some of the receivers, asks them to reveal the adjoining check vector, $B_{k,((j+1) \bmod 2)}$, and verifies $V_{k,((j+1) \bmod 2)}B_{k,((j+1) \bmod 2)} = s$. It also informs all receivers that the k'th pair is hence invalid for use.

One should note that unlike the group check vectors of [RB89], the amount of information INT is required to hold is proportional to the security parameter b. This is manifested in the length of the vectors V it holds, and reflects the

minimum number of colluding receivers which are able to recover the secret using their check vectors. It is not affected by the *potential* number of receivers, which may be higher. Also, in our protocol INT is not required to remember a different value for each receiver, and all of the check vector coordinates are used for verification by all receivers.

2.1 A Distributed Dealer

We would like to extend our protocol so that the secret V will be jointly assigned by several dealers, and such that any subset smaller than our security threshold is unable to reconstruct it. An additional requisite we have, which is driven by our voting application, is that no collusion of (a threshold of) dealers and receivers will be able to correlate the vectors V or the check vectors to the intermediaries that use them. The intuitive reason for this is that the vectors V are used in our setting as voting credentials. They are initially distributed by the dealers to voters (hence, the dealers must know who they are). However, during elections, the vectors should be used anonymously for voting.

A trivial distribution protocol that one may think of is to let the i'th dealer select the i'th coordinate of the secret V and the i'th coordinate of each of the corresponding check vectors. However, this would maintain a strong correlation between vectors and intermediaries: $log(n)$ collaborating dealers know enough coordinates of each check vector to identify it, with high probability, from a list of vote vectors received by a colluding receiver, and thus they may jeopardize voters anonymity.

In order to anonymize check vectors, we make use of an anonymous multi-party computation (AMPC) [MP01]. The idea is to generate each coordinate in V jointly by dealers and garble all vectors through AMPC so as to prevent correlating the generated value with the intermediaries that received its shares from the dealers. More precisely, recall the definition of an AMPC:

Definition 1. *[MP01]*

Let the input consist of n tuples each consisting of m values from a domain D: $\{\langle X_j^1, X_j^2, ..., X_j^m\rangle\}_{1\le j\le n} \subseteq D^m$ *. Let a function f be from domain* D^m *to some domain D'. Let* $P^1, \ldots, P^m$ *be a set of players. Suppose that each* P^i *initially knows only* $\{X_j^i\}_{1\le j\le n}$*. A protocol is an* anonymous multi party computation *of f on the inputs* $\{X_j^i\}$ *with robustness b if it calculates the set of values* $\{f(X_j^1, X_j^2, \ldots, X_j^m)\}_{1\le j\le n}$ *such that for any set of players G,* $|G| \le b$*, and any function* ϕ*, there exists a function* ψ *such that:*

$$Pr[\phi(\{f(X_j^1, X_j^2, \ldots, X_j^m)\}_{j=1..n}, AMPC^G) = (X_j^1, X_j^2, \ldots, X_j^m)] \le$$
$$Pr[\psi(\{f(X_j^1, X_j^2, \ldots, X_j^m)\}_{j=1..n}) = (X_j^1, X_j^2, \ldots, X_j^m)] \ .$$

The definition above uses $\phi(I, AMPC^G)$ to denote the computation of $\phi(I)$ using any internal values known to G during the AMPC (for any ϕ, $\phi(I)$ alone denotes the computation of ϕ without knowledge of any such internal values). For completeness, a realization of AMPC is depicted in Figure 1 in the appendix.

In our protocol, the function f is simply addition, and the m values in each tuple are the m additive shares $v_{ij}, j = 1..m$ issued by the m dealers, of the v_i coordinate in the secret vector V. For a security parameter b, we require that the minimal length p of V and the corresponding check vectors will be $b + 2$.

The multi-dealer information checking protocol is then as follows:

1. For each intermediary, the j'th dealer DLR_j issues additive shares $v_{1j}, ..., v_{pj}$ of each of the p coordinates of vector V. For purposes of correlating V with its check vector, the j'th dealer also issues an additive share vid_j of a voter-id vid.
2. The intermediary computes the coordinates of her secret vector V, $v_1, ..., v_p$, and her voter-id vid by summing up the shares received from the different dealers: $v_i = \sum_{j=1..m} v_{ij}$, $vid = \sum_{j=1..m} vid_j$.
3. We now employ an AMPC p times as follows. For each coordinate i, $1 \leq i \leq p$, and all intermediaries, the j'th dealer provides v_{ij}, vid_j as input to the j'th AMPC top player (P^j). The AMPC then produces a permuted list of i'th coordinates $v_i = \sum_{j=1..m} v_{ij}$ of all intermediaries along with their corresponding vid. This list is sent securely to an authority that we call A_i (this could be an auxiliary authority, or one of the talliers/registrars). The result of these p AMPCs is that there are p authorities $A_1, ..., A_b$ that each holds $\langle v_1, vid \rangle, ..., \langle v_p, vid \rangle$, respectively, for every intermediary, such that there is no known correlation between the pairs $\langle v_i, vid \rangle$ and the intermediaries known to the dealers. [2]
4. The p authorities now together compute check vectors for V as follows (we present the protocol for computing one check vector for one receiver; the check vectors for the rest of the receivers are computed in a similar manner).
   a) A_1 selects a random number r_1 and passes it to A_2. In addition, A_1 splits r_1 into p shares, $r_{11}, ..., r_{1p}$, such that $r_{11} + r_{12} + .. + r_{1p} = r_1$, and sends r_{1k} to A_k.
   b) From here on, the k'th authority A_k, for $2 \leq k \leq p$, selects randomly the k'th component of the check vector, b_k, as well as a random number r_k. It passes b_k, vid to the receiver. It then computes $c_k = c_{k-1} + b_k v_k + r_k$ and sends c_k to A_{k+1} (the last authority A_p passes back to A_1). In addition, A_k splits r_k into p shares, $r_{k1}, ..., r_{kp}$ and sends r_{ki} to A_i.
   c) Every A_k computes $r'_k = r_{1k} + r_{2k} + .. + r_{pk}$ and sends it to A_1. A_1 computes b_1 as follows: $b_1 = (s - (c_p - \sum r'_k))/v_1$. This computation is feasible if we are working in Z_q, for a prime q, in $O(log(q))$ time (see e.g., [CLR89]). This guarantees that $\sum b_i v_i = s$. It then sends b_1, vid to the receiver.

At the end of the enhanced check vectors protocol, each receiver RCV_j holds a list of check vector pairs $\langle vid, B \rangle$ that each corresponds to an anonymous intermediary holding $\langle vid, V \rangle$ such that $VB = s$. This protocol may be repeated

[2] To be precise, we should add intermediary-indices to vector components and vids. Thus, the AMPC is used to generate lists of values, e.g., i'th vector coordinates $\{v_i^{INT}\}_{INT}$ for all voters INT, and likewise, voter-id's $\{vid^{INT}\}_{INT}$ for all voters. For simplicity, we omit these indices from the description.

multiple times to generate a list of such pairs per vote (as indeed is done to facilitate testing and receipts).

3 Voting Protocol

3.1 Review

The voting protocol is composed of the following phases:

1. Registration - during this stage, the voters identify themselves to $b + 1$ registrars (using IDs established by other means, such as passports or national identity cards), and receive their vote vectors. A voter receives her vote vectors for each possible ballot s. We employ the enhanced check vectors protocol here, so a vote vector corresponds to the secret V that an intermediary INT (the voter) receives from the dealer (registrars), and the check vectors B for a vote vector are computed and distributed to the receivers (talliers).
2. Testing - during this stage the voter verifies (with high probability) that the vote vectors she receives from the registrar(s) would indeed enable her to vote.
3. Voting - The voter sends the vote vector V that corresponds to her desired ballot s to some of the talliers. Each of these talliers holds a corresponding check vector B for that vote vector and verifies that indeed $VB = s$. During this stage the voter is also able to verify that her vote was received properly by the talliers by checking a receipt she receives from each tallier.
4. Counting - each tallier publishes all the ballots it received, and everyone can verify the election results.

3.2 The Voting Protocol

We now describe the voting protocol in more detail.

Voter Registration. The enhanced check vectors protocol of Section 2 is employed by the group of registrars (the dealers) to issue all voters their anonymous secret vectors V, secret meaning s, secret voter-ids *vid*, and to issue corresponding anonymous check vectors to talliers. For each ballot s, each voter is issued several pairs of vectors $\{\langle V_{1,0}, V_{1,1}\rangle, \langle V_{2,0}, V_{2,1}\rangle, ...,\}$ in order to facilitate testing and receipts.

Testing. During this stage the voter verifies that talliers indeed hold a valid check vector for his vote. As mentioned, each voter holds several pairs of vectors for each candidate. She randomly selects one of the vectors, say $V_{1,0}$, and sends it to several talliers. Each tallier sends back the check vector corresponding to the adjoining vector in the pair, in our case $B_{1,1}$. The voter verifies that $V_{1,1}B_{1,1} = s$. She should also publish the vote vector pair she used for verification in order to disqualify it for use in the elections.

Voting. The voter anonymously sends one vote vector designating her ballot, along with her *vid*, to at least $b+1$ different talliers. [3] Each tallier verifies that the vote vector received, say $V_{i,1}$, satisfies $V_{i,1}B_{i,1} = s$, and s is in S. The adjoining vote vector in the pair that was used serves as the verification vector. That is, when a tallier receives a vote vector, say $V_{i,1}$, it sends back as a receipt the check vector $B_{i,0}$. The voter then verifies $V_{i,0}B_{i,0} = s$ to verify $B_{i,0}$'s authenticity.

Counting. At the end of election day each tallier publishes each vote vector it received along with its *vid*. Other talliers may verify the validity of any published vector (via check vectors) and include it in their tally even if they did not receive it directly from the voter. Each voter may also verify that her vote appears in the list of published votes. The majority's tally then determines the results of the elections.

4 Security Analysis

In this section we analyze the security properties of the electronic voting scheme, roughly according the criteria posed by Cranor and Cytron in [CC97], as detailed below.

For the formal analysis, we will make use of the following notation. We denote by n the number of voters. The number of registrars, auxiliaries and talliers is each m, out of which a total of b corrupt authorities combined of all types is assumed, where $b = \lceil \frac{m}{2} \rceil - 1$. We denote by $p = b + 2$, the typical length of vectors we employ. Finally, let q be the prime size of the finite field Z_q we operate with. Let S be the set of possible valid candidate-ballots. We will use *with high probability* to mean with probability at least $1 - |S|/q$.

4.1 Accuracy

The first requirement of a voting systems is the *accuracy* of the tally it produces. Specifically, it must disallow stuffing of ballots by any participant (including the authorities), and prevent the elimination and the modification of valid ballots.

In order for any vote V to be counted in the final tally in our scheme, there must exist $b+1$ different talliers that accept it. Hence, at least one honest tallier must include the vote in his tally. This means that there is an honest tallier for whom the check vectors B satisfy $VB = s$ for some candidate s. The question is how difficult is it to forge such V. Here, we focus on a particular vote vector V and a corresponding check vector B held by an honest tallier. We show that there is low probability that a collusion of less than $p-1$ authorities (of any combination of registrars, auxiliaries, and talliers) can forge V' for which $V'B = s'$, where s' is an acceptable ballot in S. This proves that forged or modified ballots cannot be stuffed in the ballot box. It remains to argue that valid ballots cannot be

[3] Anonymized communication can again be achieved via AMPC [MP01], so that the voter does not need to employ cryptographic software for this.

eliminated. This is simply a result of the fact that a voter receives a receipt from $b+1$ talliers.

In order to prove that ballots cannot be forged, we commence by showing that knowledge of at least p check vectors, or t check vectors and u vote vector coordinates where $t+u=p$, is required in order to find a V that will satisfy $VB=s$ for a check vector B unknown to corrupt parties. (Lemma 1). We then show that any set of less than $p-1$ servers does not have sufficient information in order to reveal an additional coordinate of B or V not known to them (Lemma 3 and 4). Together, this implies that no group of $b=p-2$ colluding authorities can find a vote vector V that satisfies the missing check vector equation.

Lemma 1. *Let B be a check vector held by some honest tallier (hence unknown to anyone but the tallier). Then w.h.p. knowledge of at least p check vectors, or t check vectors and u coordinates of V, where $t+u=p$, is required in order to find V' such that $V'B=s$.*

Proof. Since the check vectors are created independently, the only correlation between the check vectors known to the colluding (corrupt) authorities and the unknown check vector held by the honest authority, is the set of constraints $VB=s$ for all B. Thus in order to find a V' that satisfies $V'B=s'$, the colluding authorities actually have to solve for V. As all vectors are of length p, then in order to solve for V one needs to reduce sufficiently many equations. If the degree of any equation set is p, then clearly knowledge of p items (either p check vectors or t check vectors and u coordinates from the vector V which satisfies all check vector equations) is necessary for solving it.

It is left to argue that degree of the equations set is indeed p. The check vectors are created independently by the auxiliaries for each tallier, and each auxiliary controls the creation of a separate coordinate of the check vectors. Therefore, the creation process is equivalent to the repeated selection of a random vector B that satisfies $BV=s$. W.h.p. the choice of m such vectors ($m >= p$) yields a vector set for which each subset of p vectors are in the general position. Though this is a classic result, we give the intuition here: Every $k<p$ vectors span a sub-space of dimension at most $p-1$. Any k'th vector is chosen randomly from the entire p-dimensional space. Therefore, the probability that the k'th vector belongs to the $(p-1)$-dimensional subspace defined by $p-1$ other vectors is $1/q$. Thus, w.h.p, each subset of size p or less of equations $VB_i=s$ is also independent.

Lemma 2. *Let r_i be a random value used by an honest auxiliary authority in the enhanced check vectors protocol. Then with high probability, a cooperation of at least $p-1$ auxiliary authorities is required in order to reveal r_i.*

Proof. The generation of r_i in the enhanced check vectors protocol induces the following constraints:

1. $r_i=\sum_j r_{ij}$, where r_{ij} is sent to the j'th auxiliary.
2. $r'_k=\sum_k r_{kj}$, where r_{kj} is produced by the k'th authority so that $r_k=\sum_j r_{kj}$, and where all of the r'_k are known to the first auxiliary authority A_1.

A collusion of x auxiliaries, $x < p-1$, that includes A_1 knows all the r'_k's, x of the equations designated in item 2, and x of the equations designated in item 1. Since r_i is a sum of p independent variables chosen uniformly at random, in order to solve for r_i it is necessary to know all of the r_{ij}. However, from the above it is clear that a cooperation of less than $p-1$ auxiliaries has only $p-2$ equations for p variables, and hence leaves every value in Z_q a possible solution for r_i, and guessing has a probability of $|S|/q$ of success.

Lemma 3. *Let b_i be the i'th check vector coordinate chosen by an honest (i'th) auxiliary authority in the enhanced check vectors protocol and held by an honest tallier. Then a cooperation of at least $p-1$ corrupt auxiliaries is required to reveal b_i.*

Proof. The generation of b_i induces the following constraints:

1. $c_i = c_{i-1} + b_i v_i + r_i$, where c_{i-1} is produced by A_{i-1}, c_i is passes on to A_{i+1}, and v_i is known to A_i.
2. $\sum v_i b_i \in S$ (the check vector equation).

By Lemma 2 above, r_i is not known to a collusion of less than $p-1$ auxiliaries. Hence, for a collusion of less than $p-1$ there are 3 unknowns in equation 1 above and two more unknowns in Equation 2 above. Together, they leave every value in Z_q a possible solution for b_i, and guessing has a probability of $|S|/q$ of success.

Lemma 4. *Let v_i be the i'th coordinate of the vote vector V designated in the enhanced check vectors protocol. Suppose that auxiliary authority A_i holding v_i is honest. Then a cooperation of at least $p-1$ corrupt authorities is required to reveal v_i.*

Proof. The generation of v_i induces the following constraints:

1. $v_i = \sum v_{ij}$ where v_{ij} is chosen by the j'th registrar (and passed in the AMPC computation, which is secure by [MP01]).
2. $c_i = c_{i-1} + b_i v_i + r_i$, where c_{i-1} is known by A_{i-1}, c_i is passed to A_{i+1}, and b_i is known to a corresponding tallier. Note that by Lemma 2, r_i is unknown to the collusion.
3. $\sum v_i b_i = s$ (the check vector equations).

A collusion of less than $p-1$ authorities may solve for up to $p-2$ of the vote vector v_j's, as a result of vote vector coordinates known to corrupt auxiliaries and check vectors known to corrupt talliers. This still leaves all possible values for v_i. In addition, they also know at most $p-2$ values in equation 1, which also leaves all possible values for v_i. Finally, they know some equations of the form 2 above for which c_{i-1}, c_i and b_i are known but r_i is not known. Since each such equation adds an unknown r_i, they still are unable to solve for v_i.

4.2 Democracy

The second requirement of a voting system is that it preserves the "one person one vote" rule, allowing every eligible voter and only them one and only one vote. That an eligible voter receives a valid vote vector during registration is guaranteed by our Testing step. Specifically, the probability of detecting corruption at registration (e.g., by providing the voter with invalid vote vectors or some of the talliers with invalid check vector shares) can be made as high as desired by tuning the amount of pre-voting testing. The vote once rule is maintained by having each voter attach her *vid* to the vote and allow each *vid* to appear only once in the final tally. Finally, we already discussed in the accuracy analysis the possibility of producing forged votes, and concluded that the scheme cannot allow ineligible entities to vote.

4.3 Verifiability

The third requirement is that of verifiability. One version of this requirement is individual verifiability, namely that individuals are able to verify that their votes had been counted correctly. A stronger requirement is universal verifiability [BY86], ensuring that any party, including a passive observer, can convince herself that the election is fair, i.e., that the published final tally is computed fairly from the ballots that were correctly cast. Since the individual vote vectors are published by the talliers in our scheme, both individual verifiability and universal verifiability are maintained.

4.4 Privacy and Receipt Freeness

Finally, a fundamental requirement is the need to ensure voters privacy, i.e., that how an individual votes will be kept secret. A stronger requirement of privacy is that the voter is unable to prove how she voted. This requirement is important in order to prevent vote buying and voter coercion. However, this requirement is contradictory to individual verifiability - being able to verify that ones vote is counted correctly usually means also that one can prove how she voted.

In order to link ballots with voters, one actually needs to be able to correlate either vid's or vote vectors with voters. Since ballots are delivered to talliers over an anonymous network, and since we assume the communication channels between the voter and the non-corrupt entry points of the anonymous network are secure, talliers cannot trace ballots back to voters. Since each registrar knows only a share of each vote vector coordinate, any collusion of $p-1$ or fewer registrars knows nothing (in the information theoretic sense) about any vote vector, since they are still missing one share of each coordinate. auxiliary authorities, on the other hand, do know vote vector coordinates. However, they receive these as the results of an AMPC, and hence, by the properties of the AMPC [MP01] they cannot correlate them to any input share which registrars can relate to voters.

The fact that our system provides a receipt for the voter, does not necessarily mean that it is not receipt free: Since the voter (and only her), can produce 'fake' receipts for every possible ballot, a third party, to whom the voter presents her

Vote Vectors and receipts, is still unable to know to whom the voter actually voted.

However, a voter can reveal her VID, in order to prove how she voted. Thus our system is currently not receipt free.

References

[BY86] J. Benaloh and M. Yung. "Distributing the power of a government to enhance the privacy of voters". In *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 52–62, 1986.

[CIV01] California Internet Voting Task Force final report, available at `http://www.ss.ca.gov/executive/ivote/`

[Cha81] D. Chaum. "Untraceable electronic mail, return addresses and digital pseudonyms". *Communications of the ACM* 24(2):84-88, 1981.

[Cha85] D. Chaum. "Security without identification: Transaction systems to make big brother obsolete". *Communication of the ACM* 28(1):1030-1044, 1985.

[CFSY96] R. Cramer, M. Franklin, B. Schoenmakers and M. Yung. "Multi-authority secret-ballot elections with linear work". *LNCS 1070, Advances in Cryptology – EUROCRYPT '96*, pp. 72-83, 1996.

[CGS97] R. Cramer, R. Gennaro and B. Schoenmakers. "A secure and optimally efficient multi-authority election scheme". *LNCS 1233, Advances in Cryptology – EUROCRYPT '97*, pp. 103–118, 1997.

[CC97] L. F. Cranor and R. K. Cytron. "Sensus: A security-conscious electronic polling system for the Internet". *Proceedings of the Hawai'i International Conference on System Sciences*, 1997, Wailea, Hawaii.

[CLR89] T. H. Cormen, C. E. Leiserson and R. L. Rivest. "Introduction to Algorithms". MIT Press, 1989

[CM01] CALTECH-MIT/Voting Technology Project `http://www.vote.caltech.edu/Reports/index.html`

[DmLM82] R. DeMillo, N. Lynch, and M. Merritt. "Cryptographic protocols". *Proceedings of the 14th Annual Symposium on the Theory of Computing*, pp. 383-400, 1982.

[DuR99] B. W. DuRette. "Multiple administrators for electronic voting". B.Sc thesis, MIT, 1999. `http://theory.lcs.mit.edu/~cis/theses/DuRette-bachelors.pdf`

[FY94] M. Franklin and M. Yung. "The blinding of weak signatures (extended abstract)". *LNCS 950, Advances in Cryptology – EUROCRYPT 94*, pp. 67-76, 1995.

[FOO92] B. Fujioka, T. Okamoto and K. Ohta. "A practical secret voting scheme for large scale elections". *LNCS 718, Advances in Cryptology – ASIACRYPT '92*, pp. 244-251, 1992.

[HS98] Q. He and Z. Su. "A new practical secure e-voting scheme". *IFIP/SEC '98 14th International Information Security Conference*, 1998.

[IPI01] Internet Policy Institute, Report of the National Workshop on Internet Voting: Issues and research agenda. Available at: `http://www.netvoting.org/Resources/InternetVotingReport.pdf`

[MP01] D. Malkhi and E. Pavlov. "Anonymity without 'Cryptography' ". *Proceedings of Financial Cryptography '01* (FC '01).

[RB89] T. Rabin and M. Ben-Or. "Verifiable secret sharing and multiparty protocols with honest majority". In *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, pp. 73–85, 1989.

[S91] A. Salomaa. "Verifying and recasting secret ballots in computer networks". *LNCS 555, New Results and New Trends in Computer Science*, pp. 283-289, 1991.

[Sch99] B. Schoenmakers. "A Simple publicly verifiable secret sharing scheme and its application to electronic voting". *LNCS 1666, Advances in Cryptology - CRYPTO '99*, pp. 148-164, 1999.

[WALS02] M. Wright, M. Adler, B. N. Levine, C. Shields. "An Analysis of the Degradation of Anonymous Protocols". In *Proceedings of Network and Distributed System Security Symposium*, 2002.

A An AMPC Realization

For completeness, we depict here the construction of an AMPC computation [MP01]. The picture shows a network of m^2 players implementing an AMPC such that any collusion of $m-1$ corrupt players can be tolerated.

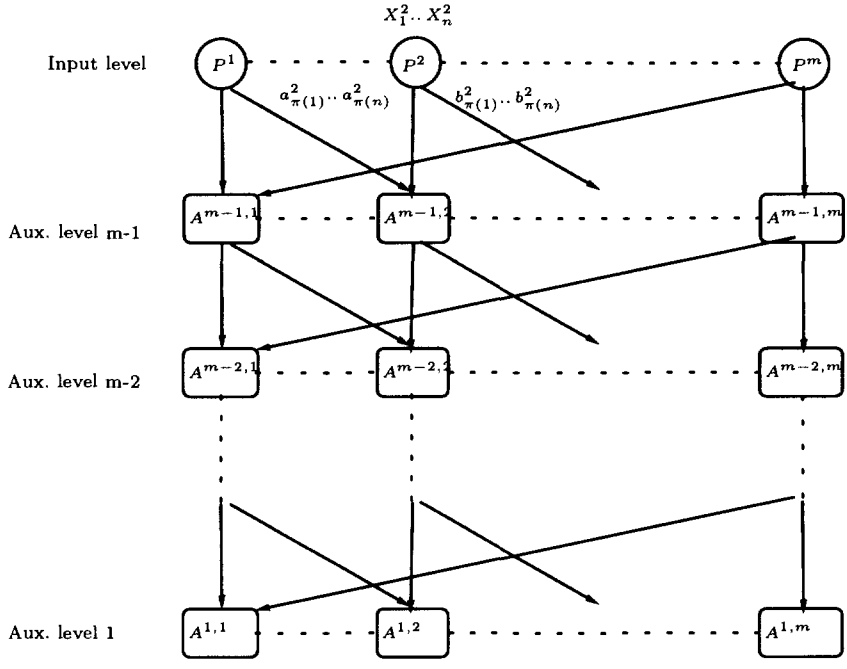


Fig. 1. The communication graph in an AMPC.

An Implementation of a Universally Verifiable Electronic Voting Scheme Based on Shuffling

Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako

Internet Systems Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae Kawasaki 216-8555 Japan
j-furukawa@ay.jp.nec.com, ke-mori@bx.jp.nec.com, obana@bx.jp.nec.com,
h-miyauchi@bc.jp.nec.com, k-sako@ab.jp.nec.com

Abstract. This paper discusses the implementation of the voting scheme based on mix-net technology. The advantages of employing this technology are that voters can vote-and-go, and that it is flexible enough to be used for variety of vote-expression methods, while ensuring the privacy of votes and the elimination of faulty players. The most attractive security feature of this scheme is its universal verifiability; anyone can confirm the correctness of the result.
Such verifiability is achieved by providing proofs on correct shuffling and decryption. The paper presents a new scheme for generating a single proof for shuffle-and-decrypt process. Compared to the combination of two separate proofs on shuffle and decryption, the new scheme is 150% faster with only 80% of the length. As a result, the system was able to produce results that were verified correct within *twenty* minutes following a vote participated in by ten thousand voters, with three shuffling centers being used.
We believe this is the first implementation report of a voting scheme with universal verifiability.

Keywords: voting, election, privacy, shuffle, zero-knowledge proof, mix-net

1 Introduction

1.1 Voting Schemes

The three basic requirements on a secure voting system are detection of faulty voters; detection of faulty centers; and vote secrecy. The result of research in cryptographic protocols gave us several variations of voting protocols achieving the above requirements, which can be grouped under the following three categories:

- Blind signature based schemes:[FOO92,Sak94,OMAFO]
- Homomorphic encryption based schemes:[CY85,SK94,CFSY96,CGS97, DJ01]
- Shuffling based schemes:[PIK93,SK95,Abe99,FS01,Ne01]

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 16–30, 2003.

Blind signature based protocol is the most computationally efficient one, and have been implemented in [SENSUS,EVOX]. However, this scheme has only individual verifiability; a voter can confirm that his own vote is accepted by the authority, but no outside 'observer' can verify the whole procedure. Secondly, this schemes requires a channel that is untraceable, which is not easy to achieve in a real life. Furthermore, the voter must engage in all phases of voting, and the existence of any voter who fails in completing a process threats a sound execution of the voting result.

Homomorphic encryption based schemes is an elegant and efficient scheme when used for yes/no-vote case. However, they suffer largely from inflexibility in representing a vote. The extension the scheme to 1-out-of L voting will sacrifice the efficiency with large L and large number of voters,and the size of L that is allowable is unknown.

On the other hand, the shuffling based schemes, also known as mix-net, bears many desirable features; it enjoys flexibility in representing any vote, the voters can simply vote-and-go and requires only a small computational ability. Moreover, by having authorities prove the correctness of their procedures, it achieves public verifiability. With this capability, we can employ 'observers', third parties to ensure the correctness of procedures. In a paper-based voting scheme, the existence of the observers is often required by law.

The cost is paid for this property. Proofs for assuring correctness of shuffling and decryption are not for free. Since proving the correctness of shuffling is more expensive than that of decryption, many schemes were proposed to improve efficiency[SK95,Abe99,FS01,Ne01]. Most of the efficiency arguments made in these documents were based on theoretical evaluations, like number of modular exponentiations. To the authors' knowledge, no voting system which uses these scheme was reported, which indicates the scheme is yet to be believed practical.

1.2 Our Contribution

We report here the first implementation of universally verifiable voting scheme based on shuffling. We present a new scheme to prove correctness of shuffle-and-decrypt process. It is, in a way, a merge of an efficient proof on shuffling proposed in [FS01] and an efficient proof on decryption using the techniques from [Br93]. Experimental results show that it is 150% faster than performing each proof separately, and a proof is 20% shorter. The proposed scheme can be proved to be correct and sound.

However, we cannot prove any zero-knowledgeness of the resulted scheme. Moreover, we found that the protocol of [FS01] which we based our protocol does not satisfy the definition of computational zero-knowledge. On the other hand, we considered security requirements regarding which information must be kept hidden in the protocol. As a result, we can prove that no polynomially bounded adversary can compute any partial information of the permutation from the protocol.

Using the newly proposed scheme, we have implemented a voting system which is publicly verifiable. Our system can output a certified tally for 10,000

voters within 20 minutes, with three shuffling centers, each on a personal computer being used.

We note that receipt-freeness is not the aim of our system, although we are aware of potential threats of vote buying and coercing due to lack of the receipt-free property.

1.3 Organization of the Paper

In Section 2, we describe a new scheme to prove correctness of shuffle-and-decrypt procedure. In Section 3 we provide implementation details of our voting system. We also provide experimental results comparing different proof scheme we have implemented.

2 The New Scheme for Proving Shuffle-and-Decrypt Procedure

In this section, we provide details of the new scheme. First, we briefly describe the shuffling based voting scheme, and discuss why we need shuffle-and-decrypt proofs.

2.1 An Overview of the Shuffling Based Voting

The shuffling based schemes, also known as mix-net, achieve anonymity of votes by distributing the decryption key among multiple authorities called shuffling centers. Voters send signed but encrypted votes. These votes will be processed by the shuffling centers, who shuffles the votes and decrypt them. After all the shuffling centers have done their work, the encrypted vote will be completely decrypted, but its original owner can not be identified due to the shuffles. By providing shuffle-and-decrypt proofs, anyone can verify the output is a result of correct decryptions of the shuffled valid votes. The proofs should be made in a way it will not reveal the shuffle nor the decryption key, so it would not infringe the vote anonymity.

2.2 How Votes Are Generated and Processed

The votes are encrypted using ElGamal cryptosystems[E85], with public keys (p, q, g, Y) and a secret key $\bar{x} \in \mathbf{Z_q}$ s.t. $Y = g^{\bar{x}} \bmod p$. Here, p and q are two primes s.t. $p = kq + 1$, where k is an integer, and g is an element that generates a subgroup $\mathbf{G_q}$ of order q in $\mathbf{Z}/p\mathbf{Z}$.

The vote m_v is encrypted to a ciphertext (G, M) where $G = g^{\bar{r}} \bmod p$ and $M = m_v \cdot Y^{\bar{r}}$, and $\bar{r}$ is a random element in $\mathbf{Z}/q\mathbf{Z}$, chosen by the voter.

Such ciphertext can be decrypted by a person knowing the secret key $\bar{x}$. The vote is recovered by: $M/G^{\bar{x}} \bmod p$. However, in the voting scheme no single entity knows $\bar{x}$. The secret key $\bar{x}$ is distributed among m shuffling

centers. Each shuffling center SC_i has a secret key $x_i \in \mathbf{Z}/q\mathbf{Z}$ and corresponding public key $y_i = g^{x_i}$. The sum of their secret keys gives $\bar{x}$. Given a ciphertext $(G, M) = (G^{(1)}, M^{(1)})$, when all the shuffling centers computes $(G^{(i+1)}, M^{(i+1)}) = (G^{(i)}, M^{(i)}/(G^{(i)})^{x_i} \bmod p)$ sequentially, then the second component of the output from the last shuffling center gives the decrypted result,$m_v = M/G^{\bar{x}} \bmod p$.

A ciphertext (G, M) can be randomly transformed to another ciphertext which decrypts to a same message. Using a randomly generated $s \in \mathbf{Z}/q\mathbf{Z}$, such transformed ciphertext, $(G \cdot g^s, M \cdot Y^s)$, is unlinkable to original ciphertext (G, M). This will be useful for shuffling ciphertext, which will be described in the next subsection.

2.3 Shuffle-and-Decrypt

For multiple votes, each shuffling center shuffles the input votes before decrypting each of them by his own secret key. We call this procedure 'shuffle-and-decrypt'. For simplicity, we concentrate on one shuffling center and denote his secret key as x. We represent by $\bar{y}$ the product of the public keys of subsequent centers.

Given n ciphertexts $\{(G_i, M_i)\}$, where all $\{G_i\}$ and $\{M_i\}$ have the order q, the shuffling center randomly chooses a permutation π and a random element $s_i \in_U \mathbf{Z}/q\mathbf{Z}$ to obtain shuffle-and-decrypt result as follows:

$$(G'_i, M'_i) = \left(g^{s_i} G_{\pi(i)}, \bar{y}^{s_i} M_{\pi(i)} / G'^{x}_i\right) \bmod p (i = 1, \ldots, n)$$

Details of this procedure is given in Subsection 3.2.

2.4 Generation of the Proof

We now provide the new scheme to generate a proof that the shuffling center (which will be denoted as the prover in the sequel) indeed shuffled and decrypted honestly. The proof adds proof of decryption on the top of proof of shuffle proposed in [FS01]. The newly added proving procedures are the equations labeled (2) to (5).

We describe the scheme in a non-interactive way, where a challenge from a verifier is given as an output of some universal one-way hash functions. We assume here the order of elements of input ciphertexts (G_i, M_i) and output ciphertexts (G'_i, M'_i) are all q.[1]

To prove (G'_i, M'_i) are generated correctly from (G_i, M_i), the prover computes the following equations for randomly chosen $z, z_i, \rho, \sigma, \tau$, λ and $\lambda_i, z' \in_U \mathbf{Z}/q\mathbf{Z}$

[1] We note here that it is important for both shuffling centers and verifiers to check that the order are indeed identically q. If one of the input ciphertexts are not of order q, then the resulting output may reveal the permutation used in the protocol. Or, the shuffling center may intentionally present wrong ordered ciphertexts which pass the verification formula but is not a correct shuffle.

$(i = 1, \ldots, n)$: We use $\mathcal{H}$ and $\tilde{\mathcal{H}}$ to denote universal one-way hash functions which output an element of $\mathbf{Z}/q\mathbf{Z}$ and $\mathbf{G_q}$, respectively.

$$\tilde{g} = \tilde{\mathcal{H}}(p, q, g, Y, 0), \quad \tilde{g}_i = \tilde{\mathcal{H}}(p, q, g, Y, i)$$
$$v = g^{\rho}, \quad w = g^{\sigma}, \quad t = g^{\tau}, \quad u = g^{\lambda}, \quad u_i = g^{\lambda_i} \bmod p$$
$$\tilde{g}'_i = \tilde{g}^{s_i}\tilde{g}_{\pi(i)}, \qquad \tilde{g}' = \tilde{g}^{z}\prod_{j=1}^{n}\tilde{g}_j{}^{z_j} \bmod p$$
$$g' = g^{z}\prod_{j=1}^{n}G_j{}^{z_j}, \qquad m' = \bar{y}^{z}\prod_{j=1}^{n}M_j{}^{z_j} \bmod p$$
$$\dot{t}_i = g^{3z_{\pi(i)}+\tau\lambda_i}, \qquad \dot{v}_i = g^{3z_{\pi(i)}{}^2+\rho s_i} \bmod p$$
$$\dot{v} = g^{\sum_{j=1}^{n} z_j{}^3+\tau\lambda+\rho z} \bmod p$$
$$\dot{w}_i = g^{2z_{\pi(i)}+\sigma s_i}, \qquad \dot{w} = g^{\sum_{j=1}^{n} z_j{}^2+\sigma z} \bmod p$$
$$c_i = \mathcal{H}(p, q, g, \bar{y}, \tilde{g}, \{\tilde{g}_j\}, \{(G_j, M_j)\}, \{(G'_j, M'_j)\},$$
$$\tilde{g}', \{\tilde{g}'_j\}, g', m', v, w, t, u, \{u_j\},$$
$$\{\dot{t}_j\}, \dot{v}, \{\dot{v}_j\}, \dot{w}, \{\dot{w}_j\}, i : (j = 1, \ldots, n)) \tag{1}$$
$$r_i = c_{\pi^{-1}(i)} + z_i, \qquad r = \sum_{j=1}^{n} s_j c_j + z \bmod q$$
$$\lambda' = \sum_{j=1}^{n}\lambda_j c_j{}^2 + \lambda \bmod q$$
$$\zeta = \prod_{j=1}^{n} G'_j{}^{c_j}, \qquad \eta = \zeta^{x} \bmod p \tag{2}$$
$$y' = g^{z'}, \qquad \eta' = \zeta^{z'} \bmod p \tag{3}$$
$$c' = \mathcal{H}(p, q, g, y, \zeta, \eta, y', \eta') \tag{4}$$
$$r' = c'x + z' \bmod q \tag{5}$$

The prover send the proof $g', m', \tilde{g}', \tilde{g}'_i, v, w, t, u, u_i, \dot{t}_i, \dot{v}_i, \dot{v}, \dot{w}_i, \dot{w}, r, r_i, \lambda', \eta, \eta', y', r'$ $(i = 1, \ldots, n)$ to the verifier along with $\{(G'_i, M'_i)\}$.

2.5 Verifications of the Proof

The verifier first computes $(c_i)_{(i=1,\ldots,n)}$ according to eq.(1). Next, the verifier compute

$$\zeta = \prod_{j=1}^{n} G'_j{}^{c_j} \bmod p$$

and generate c' according to eq.(4).

The verifier accepts the proof if all of the following equations hold.

$$
\begin{aligned}
v^q = t^q = w^q &= 1 \bmod p \\
g^r \prod_{j=1}^{n} {G_j}^{r_j} &= g'\zeta \bmod p \\
\bar{y}^r \prod_{j=1}^{n} {M_j}^{r_j} &= \eta\, m' \prod_{j=1}^{n} {M'_j}^{c_j} \bmod p \\
\tilde{g}^r \prod_{j=1}^{n} \tilde{g}_j{}^{r_j} &= \tilde{g}' \prod_{j=1}^{n} \tilde{g}'_j{}^{c_j} \bmod p \\
g^{\lambda'} &= u \prod_{j=1}^{n} {u_j}^{c_j^2} \bmod p \\
t^{\lambda'} v^r g^{\sum_{j=1}^{n}({r_j}^3 - {c_j}^3)} &= \dot{v} \prod_{j=1}^{n} \dot{t}_j{}^{{c_j}^2} \prod_{j=1}^{n} \dot{v}_j{}^{c_j} \bmod p \\
w^r g^{\sum_{j=1}^{n}({r_j}^2 - {c_j}^2)} &= \dot{w} \prod_{j=1}^{n} \dot{w}_j{}^{c_j} \bmod p \\
g^{r'} = y^{c'} y' \ &, \ \zeta^{r'} = \eta^{c'} \eta' \bmod p.
\end{aligned}
$$

2.6 Structure of the Scheme

We briefly sketch the structure of our scheme. Our scheme is constructed by merging shuffle-proof proposed in [FS01] and decryption-proof. For the detail description of shuffle-proof, please refer to [FS01]. The decryption-proof can be constructed by well known technique to prove the equality of discrete logarithm as follows.

The prover wants to prove that M'_i is the valid decryption of $(G'_i, \bar{M}_i)$ with a private key x, that is, a relation $M'_i/\bar{M}_i = {G'_i}^x \bmod p$ holds for all i. The prover computes the followings:

$$
\begin{aligned}
c_i &= \mathcal{H}(p, q, g, y, G'_i, \bar{M}_i, M'_i : (i = 1, \ldots, n)) \\
\zeta &= \prod_{j=1}^{n} {G'_j}^{c_j} \ , \ \eta = \zeta^x \bmod p \\
y' &= g^{z'} \ , \ \eta' = \zeta^{z'} \bmod p \\
c' &= \mathcal{H}(p, q, g, y, \zeta, \eta, y', \eta') \\
r' &= c'x + z' \bmod q
\end{aligned}
$$

Then, the prover sends to the verifier, y', η', r' as a proof.

The verifier computes

$$
c_i = \mathcal{H}(p, q, g, y, G'_i, \bar{M}_i, M'_i : (i = 1, \ldots, n))
$$

$$\zeta = \prod_{j=1}^{n} G'_j{}^{c_j} \ , \ \eta = \prod_{j=1}^{n} (M'_j/\bar{M}_j)^{c_j} \bmod p$$
$$c' = \mathcal{H}(p, q, g, y, \zeta, \eta, y', \eta').$$

Then, verifier accepts the proof if the following equations hold.

$$g^{r'} = y^{c'} y' \ , \ \zeta^{r'} = \eta^{c'} \eta' \bmod p.$$

We have noticed that some of the computations done in the shuffle-proof [FS01] and in the decryption-proof are similar. For example, even though c_i is generated in a slightly different way, $\zeta = \prod_{j=1}^{n} G'_j{}^{c_j} \bmod p$ is also computed in the shuffle-proof, $\eta = \prod_{j=1}^{n} (M'_j/\bar{M}_j)^{c_j} = \prod_{j=1}^{n} M'_j{}^{c_j} / \prod_{j=1}^{n} \bar{M}_j{}^{c_j} \bmod p$ in decryption-proof and $\prod_{j=1}^{n} \bar{M}_j{}^{c_j} \bmod p$ in shuffle-proof have same factor.

We merged the two protocols in a following way: Two proof share the same c_i so that both prover and verifier need to computes above mentioned value only once. As a result of merging, experimental results show that it is 150% faster than performing each proof separately, and a proof is 20% shorter.

2.7 Properties of the Scheme

The proposed scheme is complete. Assuming that the prover is computationally bounded, i.e. solving discrete logarithm problem is hard for the prover, the scheme is sound. We claim that the protocol does not reveal the permutation; if an adversary can compute any part of the permutation from the proof, then he can solve the decision Diffie-Hellman problem.

The shuffle-proof presented in [FS01] is claimed to be computational zero-knowledge. However it is not correct.[2] However, it does not straightforwardly mean that the protocol of [FS01] leaks some information. Therefore, we considered other means to ensure the security of the protocol. What we need to ensure is that the protocol does not leak any information regarding the permutation. Therefore we formalized the situation that the protocol does not leak such information and proved that the protocol of [FS01] satisfies the definition.

Definition : Let X be a set of input and output ciphertexts, $\Pi(X)$ be a set of corresponding valid permutations, $View_{P,V}$ be a view of a verifier, that is, a random tape of the verifier and messages the verifier receives from a prover. We say the protocol (P, V) does not leak any information about the permutation if and only if:
for every polynomially bounded adversary E, there exists a polynomially bounded

[2] In the proof of Theorem 10 in [FS01], it is claimed that the probability of the distinguisher who can distinguish between the real transcript and a simulated transcript is equal to distinguishing two decision-Diffie Hellman type distributions, when we take distribution over random input. However, in the definition of computational zero-knowledge, it should be argued in the case where the input is fixed. Apparently, if the input is fixed, no similar reduction will be established.

algorithm M, such that for every polynomial $p(\cdot)$, the following inequality holds

$$\Pr[E(View_{P,V}) \in \Pi(X)] < \Pr[M(X_n) \in \Pi(X)] + \frac{1}{p(|X|)}.$$

Claim x: The protocol of [FS01] does not leak any information about the permutation.

Claim y: The proposed protocol does not leak any information about the permutation.

Proofs of claims will be presented in the final version.

3 Implementation and Its Results

We built a voting system based on the above technology. We first explain the model we employed in the system. The clarification of the responsibilities of the entities involved was quite indispensable for designing the system. It enabled us eliminate redundant computation. For example, theoretically frauds can be detected if all the centers performed verification process at every occasion. By employing a center who is responsible for verification, we can reduce the number of verifications and still have a secure system.

Then we give the procedures each entities follow. We tried to explain them in detail, including side trivial issues which is often omitted from theoretical papers whose purpose is mainly to describe new ideas.

3.1 Model

We involve five kinds of players, which are

1. Election policy committee
2. Voting center
3. Shuffling management center
4. Shuffling center
5. Voters

The election policy committee will be responsible for any fraud caused by the voting center, the shuffling management center, and the shuffling centers. The election policy committee does not engage in actual run of the electronic voting. It takes part in determining election and security policies, and assignment of the centers. The committee authorizes the output computed by the other centers, such as the parameters determined in a set-up phase and the final tally.

The voting center is in charge of handling transactions with the voters. It will announce the voting procedures, collects pro forma votes from the authorized voters, issues receipts of the collected votes, and announces the result of the tally. The voting center will receive the result of the tally by sending the list of collected votes to the shuffling management center.

The shuffling management center is responsible for decrypting and tallying the list sent from the voting center, in collaboration with the shuffling centers. The shuffling management center passes the list to the first shuffling center, and collects his answer which will be sent to of the next shuffling center, and repeats the process until all the assigned shuffling centers shuffle and decrypt the list. The shuffled result of decryption will be sent back to the voting center. The shuffling management center is also responsible for composing a public key in the set-up phase, again in collaboration with the shuffling centers.

The shuffling center is responsible for secure management of the secret key generated in the set-up phase, and conducting decryption using the key. He is also responsible for randomly shuffling the list and keeping the shuffle confidential.

We assume all entities has his public key and corresponding secret key assured by some public key infrastructure, and the secret key is securely managed by the individual.

We require for the universal verifiability, that any party can verify that all centers conducted correctly based on the policy approved by the election policy committee. Our goal in vote privacy is that it will not be infringed as long as at least one shuffling center remains honest.

Figure 1 illustrates how these players constitute an voting system. We note that the roles of the voting center and the shuffling management center can be played by one entity.

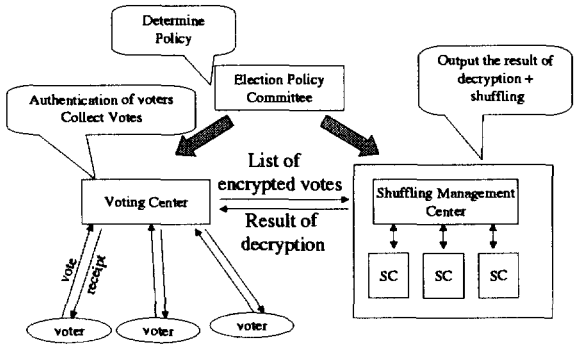


Fig. 1. System Configuration

3.2 Protocol

In this subsection we describe the procedure to set-up, the algorithm to encrypt votes, procedures to tally the votes and the algorithm to prove the correctness of the output. The protocol follows the lines of [SK95] with some minor changes for additional security. This protocol resolves the flaw in [SK95] pointed out by [MH96].

In the sequel, we assume there are m shuffling centers and n voters. All the communication between the entities are digitally signed based on a public key infrastructure.

Set-Up

1. The election policy committee will determine the parameters (p, q, g) which will be used in ElGamal cryptosystem. The numbers p and q are prime numbers satisfying $p = kq + 1$ for some integer k, and g is a generator of a group of order q on mod p.
   The shuffling management center announces the authorized parameters (p, q, g) to all the shuffling centers. The j-th shuffling center, SC_j, will randomly choose $x_j \bmod q$ as his secret key, and report his public key $y_j = g^{x_j} \bmod p$ to the shuffling management center. The report will be accompanied by the proof y'_j, r_j which ensures that SC_j indeed knows the secret x_j corresponding to y_j. The proof will be generated by SC_j as follows.

$$\begin{aligned} y'_j &= g^{\beta_j} \bmod p \\ \mathrm{c}_j &= \mathcal{H}(p, q, g, y_j, y'_j) \\ r_j &= \mathrm{c}_j x_j + \beta_j \bmod q \end{aligned}$$

   with a randomly generated $\beta_j \in \mathbf{Z}/q\mathbf{Z}$.
2. The shuffling management center will verify the proof y'_j, r_j for each public key $y_j (j = 1, \cdots, m)$ as follows.

$$\begin{aligned} \mathrm{c}_j &= \mathcal{H}(p, q, g, y_j, y'_j) \\ g^{r_j} {y_j}^{-\mathrm{c}_j} &= y'_j \bmod p \\ {y_j}^q = 1 \bmod p \ , \ y_j &\neq 1 \bmod p \end{aligned}$$

   The verified public keys are combined to compose the common public key Y.

$$Y = \prod_{j=1}^{m} y_j \bmod p$$

   The proof for each public key is necessary to ensure that the common public key Y is not generated under a control of some shuffling centers.
3. The election policy committee will certify the public keys y_j and Y properly generated as above.

Encryption of Votes

The $Voter_i$ will use the parameters Y and (p, q, g) certified by the election policy committee and encrypt his vote m_i as follows. (We assume here that m_i is selected to be an element of order q.)

$$(G_i, M_i) = (g^{\bar{r}_i}, m_i Y^{\bar{r}_i}) \bmod p$$

where $\bar{r}_i$ is an element randomly chosen by the $Voter_i$, and ID_i is information that identifies the voter. He then proves the knowledge of m_i by generating the proof α_i, t_i by

$$\begin{aligned} \alpha_i &= g^{\gamma_i} \bmod p \\ c_i &= \mathcal{H}(p, q, g, Y, G_i, \alpha_i, ID_i) \\ t_i &= c_i \bar{r}_i + \gamma_i \bmod q \end{aligned}$$

with a randomly generated γ_i. This proofs ensures that the voter who knows the content of the vote has generated the vote; a vote duplication attack by copying someone else's encrypted vote will be thwarted here.

The voting center will verify the signature and make sure that the sender is a registered voter and has not voted before. Then it will verify that the proof satisfies

$$\begin{aligned} c_i &= \mathcal{H}(p, q, g, Y, G_i, \alpha_i, ID_i) \\ g^{t_i} G_i^{-c_i} &= \alpha_i \bmod p \end{aligned}$$

and that the elements G_i and M_i are both of order q. If everything is verified, then it can optionally send back a receipt of acceptance. Such a receipt cuts in two ways: it will add confidence to the voter that the center indeed accepted his vote and will be an evidence for any disputes on vote delivery. On the other hand, it will serve as a receipt in vote-buying or coercing scenario.

Tallying

The voting center will send the list $\{(G_i, M_i)\}_{(i=1,\cdots,n)}$ to the shuffling management center. The shuffling management center will verify that all of each component is of order q, and rename them to be $(G_i, M_i) = (G_i^{(1)}, M_i^{(1)})$ for all i, which will be the input to the first shuffling center $SC1$.

The list $\{(G_i^{(j)}, M_i^{(j)})\}_i$ will be sent to SC_j. His response will be verified by the shuffling management center and will be renamed to $\{(G_i^{(j+1)}, M_i^{(j+1)})\}_i$ and sent to the next shuffling center. The response from the last shuffling center, SC_m will be verified and sent back to the voting center .

Below, we describe the procedures of each shuffling center.

1. SC_j will receive the list $\{(G_i^{(j)}, M_i^{(j)})\}_i$. He will choose a random permutation $\pi^{(j)}$ and permute the input list $\{(G_i^{(j)}, M_i^{(j)})\}_i$ and achieve the list $\{(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})\}_i$ as follows:

$$\{(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})\} = \{(G_{\pi^{(j)}(i)}^{(j)}, M_{\pi^{(j)}(i)}^{(j)})\}$$

2. The above permutation only changes the order of the ciphertexts, so it is easy to trace the permutation. In order to hide the permutation, we need to change the *look* of the ciphertext. The following procedure changes the look without changing the message hidden in the ciphertext.

First, SC_j combines the public keys of the subsequent shuffling centers as

$$Y_j = \prod_{\ell=j}^{m} y_\ell \bmod p.$$

For each of $(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})$, he chooses a random element $s_i^{(j)} \bmod q$ and obtains $\{(G'^{(j)}_i, M'^{(j)}_i)\}_i$ by

$$G'^{(j)}_i = \bar{G}_i^{(j)} \cdot g^{s_i^{(j)}} \bmod p$$
$$M'^{(j)}_i = \bar{M}_i^{(j)} \cdot Y_j^{s_i^{(j)}} \bmod p.$$

3. SC_j will decrypt each of $(G'^{(j)}_i, M'^{(j)}_i)$ using his secret key x_j as follows:

$$M''^{(j)}_i = M'^{(j)}_i / (G'^{(j)}_i)^{x_j} \bmod p \quad G''^{(j)}_i = G'^{(j)}_i$$

The list $(G''^{(j)}_i, M''^{(j)}_i)_i$ will be returned to the shuffling management center.

3.3 Proving Correctness

We employ the scheme we depicted in Section 2 to prove the correctness of the procedures performed by the shuffling centers. This proof makes the scheme universally verifiable.

In order to save time, when a shuffling center sends back the shuffled and decrypted list with the proofs, the shuffling management center first verifies the signature of the shuffling center and then pass the list to the next shuffling center. While the next shuffling center is executing shuffle, decrypt and prove procedures, the management center verifies the proof of the previous shuffling center. If the management center finds inconsistency in the proof, the procedure stops there and the faulty shuffling center is blamed. How to resume the tally will be of a human decision, taking into the account the cause of the error.

3.4 Computational Tricks

The most costly computation in the protocol is modular exponentiation. Although we have employed a quite efficient proof algorithm, it still requires the number of modular exponentiations which is in linear to the number of voters. However, looking closely at the algorithm, the most exponentiation are computed on the same base g or multiple exponentiation. Thus, We took advantage of this situation and succeeded in speeding up the system. The methods we chose are a variant of "Fixed-base comb method" and a variant of "Simultaneous multiple exponentiation method" exposed in [MOV].

3.5 Result of Implementation

We have evaluated the system under the following conditions:

- Key Size $(|p|, |q|) = (1024, 160)$.
- Security parameter k=160
- The number of shuffling centers $m = 3$.
- CPU:Athlon 1GHz, memory 256Mbyte for each of shuffling centers and shuffling management center.
- Communication Line 100baseTX

As a result, with a ten thousand voters the system can output a certified tally in 20 minutes and with a hundred thousand voters within 224 minutes, including data transmission time. Two-fifths of the time are required for computing the tally, and the rest of the time is devoted to proving and verifying the proof.

Table 1. Processing time and proof size

	Total Time		Length of Proof	
number of voters	10,000	100,000	10,000	100,000
Only Tallying (No proof)	8 min	1 hrs 10 min	–	–
New Scheme	20 min	3 hrs 44 min	12.6Mbyte	126Mbyte
two proofs with [FS01]	27 min	4 hrs 40 min	15Mbyte	150Mbyte
two proofs with [SK95]	5 hrs 20 min	(53 hrs)	865Mbyte	(8.7 Gbyte)
two proofs with [Abe99]	(9 hrs)	(120 hrs)	(545 Mbyte)	(5.2 Gbyte)
two proofs with [Ne01]	(1 hrs 20 min)	(15 hrs)	(58 Mbyte)	(580 Mbyte)

For comparison, we give in Table 1 our result on the alternative implementations, where we used two separate proofs for shuffling and decryption. For shuffling, we give two types of proofs. One is matrix based shuffling proof in [FS01], and the other cut&choose based shuffle proof described in [SK95]. We also provide estimations of the shuffle proof described in [Abe99] and [Ne01]. Those which is shown in parenthesis are based on estimation.

The author of [Ne01] claims that the scheme of [Ne01] requires only $8n$ modular exponentiations compared to $18n$ of [FS01] for the shuffle of n ciphertexts. However, in the latter it includes the cost of the both proving and verifying of shuffle, whereas the former includes only the cost for proving a general k-shuffle. We estimate that the total cost of protocol of [Ne01], including the verification, would be $47n$.

As can be seen from the Table 1, the extra time needed for providing and verifying proof of the new scheme is 12 minutes. Since the same time using two separate proofs of shuffling [FS01] and decryption takes 19 minutes, we can see the speed up of 150%. As for the size of proof, the merged new proof requires only 12.6 Mbytes whereas the two separate proofs requires 15Mbytes.

It is worth noting that although the number of exponentiations required in the scheme of [Abe99] is smaller than that of [SK95], actual time necessary to complete the protocol does not compliant. This is because the scheme of [SK95] uses a large number of fix-based computation, and merits largely from the table lookup method.

4 Concluding Remarks

In this paper, we discussed on the implementation of shuffling based voting scheme, which realizes many plausible features. We also proposed a proof that proves correctness of shuffle-and-decrypt, which is effectively faster and shorter than proving them separately. We demonstrated that an election with 100,000 voters can output the certificated result within 4 hours.

Although we currently believe that shuffling based voting scheme has many advantages over other schemes, we do not believe that this is the only solution. The privacy provided by the scheme (and also by homomorphic encryption based scheme) is conditional, that it is based on a model that the shuffling centers would not collude to intrude privacy. On the other hand, blind signature based schemes provides perfect secrecy if a voter can have an anonymous channel. Better solutions are always in search for.

References

[Abe99] M. Abe: "Mix-networks on permutation networks," *Advances in Cryptology — ASIACRYPT '99*, pp. 258–273, Springer-Verlag, 1999.

[Br93] S. Brands, *An Efficient Off-line Electronic Cash System Based On The Representation Problem*, CWI Technical Report CS-R9323, (1993)

[Cha81] D. Chaum: "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, pp. 84–88, ACM, 1981.

[CY85] J. D. Cohen and M. Yung: "Distributing the power of a government to enhance the privacy of voters," *Annual Symposium on Principles of Distributed Computing*, pp. 52–62, 1985.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers: "A secure and optimally efficient multi-authority election scheme," *European Transactions on Telecommunications*, 8:481–489, 1997. Preliminary version in *Advances in Cryptology — EUROCRYPT '97.*

[CFSY96] R. Cramer, M. Franklin, B. Schoenmakers and M. Yung: "Secure Secret Ballot Election Schemes with Linear Work," in *Advances in Cryptology — EUROCRYPT '96.*

[DJ01] I. Damgård and M. Jurik: "A Generalization, a Simplification and some Applications of Paillier's Probabilistic Public-Key system," in *Proc. of Public Key Cryptography 2001.*

[E85] T. ElGamal: "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithm," *IEEE Transactions on Information Theory*, Vol. IT-31, pp. 469–472, 1985.

[EVOX] http://theory.lcs.mit.edu/~cis/voting/voting.html

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta: "A Practical Secret Voting Scheme for Large Scale Elections," *Advances in Cryptology - AUSCRYPT '92*, pp. 244–251, Springer-Verlag, 1992.

[FS86] A. Fiat and A. Shamir. "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1986.

[FS01] J. Furukawa and K. Sako: "An Efficient Scheme for Proving an Shuffle" *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, 2001.

[MOV] Menezes, van Oorschot, Vanstone: "Handbook of Applied Cryptography" CRC Press.

[MH96] M. Michels and P. Horster: "Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme," *Advances in Cryptology - ASIACRYPT '96*, pp. 125–132, Springer-Verlag, 1996.

[Ne01] C.A. Neff, "A Verifiable Secret Shuffle and its Application to E-Voting", ACMCCS 01 pp. 116-125 2001.

[OMAFO] M. Ookubo, F. Miura, M. Abe A. Fujioka and T. Okamoto: "An improvement of a practical secret voting scheme" Information Security Workshop 1999.

[PIK93] C. Park, K. Itoh, and K. Kurosawa: "Efficient Anonymous Channel and All/Nothing Election Scheme," *Advances in Cryptology - EUROCRYPT '93*, pp. 248–259, Springer-Verlag, 1993.

[Sak94] K. Sako: "Electronic voting schemes allowing open objection to the tally," *Transactions of IEICE*, vol. E77-A No.1, Jan. 1994.

[SK94] K. Sako, J. Kilian: "Secure Voting using Partially Compatible Homomorphisms," *Advances in Cryptology - CRYPTO '94*, pp. 411–424, Springer-Verlag, 1994.

[SK95] K. Sako, J. Kilian: "Receipt-Free Mix-Type Voting Scheme," *Advances in Cryptology - EUROCRYPT'95*, pp. 393–403, Springer-Verlag, 1995.

[Schnorr] C. P. Schnorr: "Efficient signature generation by smart cards," *Journal of Cryptology*, 4, pp. 161–174, 1991.

[SENSUS] http://www.ccrc.wustl.edu/~lorracks/sensus/

Financial Instruments in Recommendation Mechanisms

Markus Jakobsson

RSA Laboratories,
Bedford, MA 01730
http://www.markus-jakobsson.com

Abstract. We demonstrate how to use financial instruments to produce recommendation mechanisms. We describe how futures and futures options, both relating to the *perception* of a company or service, can be used to derive accurate recommendations that are secure against abuse. We suggest the notion of *economic reductions* to attribute a cost to the introduction of bias in the recommendation system. We demonstrate the use of such an approach using a simplified set of assumptions on the behavior of the market.

Keywords: futures, recommendations.

1 Introduction

Mary wants to buy cashmere shocks, and has found two companies on the web that carries them. However, she has not heard of either company, and would like to know which, if any, she should make her purchase from.

This example, along with the recent expression *"On the Internet, nobody knows that you're a dog"* [6] capture – in a nutshell – the lack of trust associated with large decentralized networks. Common wisdom has it that this lack of trust is inherent, that is, that it cannot be overcome without imposing a strong structure on the network. Such a structure, however, is almost guaranteed to quench many smaller initiatives in favor of large brand-named on-line institutions, given the likely lack of time and resources to fairly assess all but the largest choices. (In our example, the mom-and-pop cashmere socks store would not be likely to be reviewed by a centralized authority, whereas a large competitor would be likely to.) Interestingly enough, it appears that the same result also is being achieved by the *lack of* such a structure, likely to be caused by the fact that consumers only trust organizations that they have already heard reassuring things about. In other words, the lack of a structure would give large organizations with a name-brand recognition among consumers a definitive edge over smaller organizations, whether these have better products or not. The question we attempt to answer in this paper is how to structure light-weight and accurate mechanisms allowing consumers to assess the value of to them unknown services of varying market penetration, and in a way that is not vulnerable to abuse. In particular,

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 31–43, 2003.

we want to avoid that one organization "cooks" the ratings in its favor (which is a security concern) at the same time as we want changes in service to be quickly reflected in the ratings (this requirement prevents static recommendations – this can be seen to be a requirement that aggrevates the design of secure solutions.) Our proposed solution is based on the rational behavior of investors by extracting a recommendation from trends in investment patterns. While markets are known not to be fully rational at all times, one can see that investors will be incentivised to detect incorrect recommendations and correct these by providing upward or downward pressure on the corresponding commodity. If we revert to our example for a moment, Mary would decide what company to deal with by inspecting stock prices measuring the quality of service of the two companies. (Note that this is different from the normal stocks, which measure the money-making abilities of the companies.) If Mary realizes that the service she obtains (after making a decision) is not consistent with the recommendation she saw, then there is an opportunity for her to make money on knowing the true value of the "goodness stock". In particular, if the socks are much better than suggested by the recommendation, then it is likely that more people soon will find out, and the stock prices measuring the sock quality will soon go up.

There are several concerns to be addressed. First and foremost, the recommendation mechanism must be *abuse-free* in that it must eventually reflect the impressions of users and buyers, and not allow deceitful bias to be introduced by companies with a stake in the outcome of the recommendation. As such, the system should defend against the effects of companies attempting to downgrade a competitor's image. A good system should, for the same reason, also guard against companies boosting their own images in ways that does not involve improving their services and products. However, neither of these requirements can be expected to be met in *full*, as suggested by the role advertisement plays in shifting the public perception in a favorable way, but without improving services and products *per se*. Therefore, our aim is to develop mechanisms that achieve a protection *strong enough* that attempts at manipulating the recommendation mechanism are less effective and more costly than other ways of getting improved ratings, such as advertisements and improved service. Analogous to how reductions provide hardness relationships in complexity theory, one can (given the right model of the market) perform *economic reductions* to demonstrate the robustness of a recommendation mechanism, by attributing a cost to the effort of manipulation. Instead of computational hardness assumptions, these would be based on economic assumptions. We draft some reasonable assumptions in order to examplify the technique and provide a rough analysis of the scheme. However, in order for a commercially meaningful reduction to be performed, much more careful modelling is required.

A second requirement on a recommendation mechanism is that it should reflect consumer opinions in a *timely* manner, that is, the recommendation should be representative of recent performance and perception. It can easily be seen that there is a conflict between the degree to which a mechanism obtains abuse-freeness, and the timeliness of the mechanism. In particular, if a mechanism only

takes the last opinion into consideration, it is easily manipulated; on the other hand, if it averages opinions over too long of a time, there is a risk that the trends are not clearly distinguishable.

Our method draws on the ecomomic incentives of investors to combine a consciousness of trends with a memory of the past, by extracting recommendations from investment statistics. It rewards investors with good foresight and punishes mistaken investors, the latter of which translates manipulative behavior into economic losses. Of course, it is important to remember that profits as well as losses would be restricted to people willing to put their money at stake, while recommendations can be provided to *anybody*.

Finally, a third issue of importance is the cost of maintaining the service. It can be seen that the costs of our mechanism are close to negligible, and its operation light-weight in that it does not require any noticeable maintenance effort. (This is the case since its costs are defrayed by the trading fees.)

Outline. We begin (in section 2) by reviewing a host of traditional solutions used for the purpose of consumer feedback. This descripton is interleaved with discussions of weaknesses of and requirements on these solutions, were they to be employed in a setting such as the Internet.

We explain the intuition of our solution in section 3. Then, in section 4, we review the structure of some financial instruments underlying our solution. In particular, we will discuss the principles behind stocks, futures, and futures options. A reader familiar with these financial primitives can go directly to the next section (section 5), in which we present our protocols for generating, evaluating and presenting recommendations. Our method relies on the principle that in a free market, the price of each commodity corresponds to the common understanding of its value. By letting commodities track aspects of companies and services that we wish to rank in the recommendation mechanism, we can simply use the ranking of their respective market values for the recommendation. Both individuals and institutional brokers may invest money in such a "market of opinions" where their investments are then translated into recommendations.

Our mechanism is therefore related to the Iowa Electronic Markets [4], but with a recommendation engine placed on top. While the Iowa Electronic Markets failed to project the outcome of the year 2000 elections – one of the markets suggested a republican victory, while the other a democratic – the mechanism still bears promise. Possible problems causing the failed prognosis may be the the closeness of the race, and that politics "infected" the game. Another problem may be the limited size of the markets. (Which suggests that the precision of our recommendation mechanism depends on the associated market size.) We refer to [2] for a more thorough discussion of these issues.

In section 6, we discuss possible user interfaces to be built on top of the recommendation mechanisms. We mostly consider the user interfaces for the "common user" (as opposed to the corporate investor.)

In section 7, we analyse the quality of our solution by providing bounds on its accuracy, and study – under a set of working assumptions we establish – the

cost of maintaining artificially high (or low) recommendations by manipulating the system.

2 Existing Solutions

We will here discuss existing mechanisms for recommendations, employed in "the real world", and explain the weaknesses of these mechanisms, were they to be employed in an Internet setting. We will, in particular, discuss the trust requirements and the risk for abuse in these schemes, along with the timeliness and the cost of collecting and maintaining the feedback.

No Recommendation Mechanism. A system not explictly employing recommendation mechanisms has to rely on advertisments and methods for retaining customers. Due to the lack of trust, it benefits already name-branded players, whether in their "known business" or when entering a new niche.

Personal Recommendations. In a system relying on *personal* recommendations (without any centralized control of these), a user may either choose only to take into consideration recommendations from to him known users (potentially using a friend-of-a-friend chain consisting of a few links); or to consider recommendations from larger sets of people. The former option suffers from a likely shortage of recommendation material (at least relating to merchants that are not household names) and an associated lack of timeliness, while the latter allows abuse. In order to curtail abuse, one can imagine a tiered recommendation structure, where users can post feedback not only about services, but also about users submitting recommendations. However, this mechanism is still exposed to abuse, and may cause "mutual over-rating" (as seen in the structure for rating buyers/sellers in eBay's system [3].) Finally, it requires mechanisms for compiling large amounts of feedback and extracting the essence of these.

Better Business Bureau. The Better Business Bureau (BBB, [1]) compiles complaints, evaluates (to some extent) the validity of these, and posts warnings when thresholds are reached. One could imagine a service of this type that not only handles negative feedback and warnings, but also postive feedback and suggestions. In either case, though, this type of structure is fraught with the problem of biased feedback. Moreover, the overhead for evaluating the feedback (and its veracity in particular) may be substantial. There is a relationship between the cost of producing a recommendation, and its accuracy and timeliness.

Reviews. A review (such as [5]) is based on surveys or feedback, and have a functional structure similar to the suggestions by a BBB. Not surprisingly, reviews suffer the same shortcomings as structures based on a BBB, but may additionally suffer from problems relating to trust. Namely, users would have to trust not only the veracity of feedback underlying the recommendation, but also the lack of bias introduced by the reviewer, particularly so of it is not clear

from where the reviewing organization receives its funding. To some extent, the quality of recommendation mechanism based on reviews depends on the number of independent reviews, and on the quantity of feedback from users. The Zagat [7] restaurant guide is a noteworthy example of a recommendation mechanism that has gained enough momentum – both among reviewers and users – to gain trust with these. Current review-based recommendation systems typically charge the user for access to recommendations. The cost of producing a recommendation relates closely to its accuracy and timeliness.

3 Intuition

In order to achieve our goals, we will take the novel approach of employing financial instruments to extract recommendations. Thus, recommendations will be based on the current market value of *opinions* about a company or service (as opposed to the company stock in general). The recommendations come with the implicit guarantee that any measureable error in the recommendation (and its timeliness) corresponds to a financial opportunity for anybody who discovers this fact. This will serve to quickly correct recommendations and to keep them as honest as can be.

Our mechanism is secure against "biased buying" by parties interested in thwarting the recommendation outcome. This follows from the fact that the economic power of any company is miniscule in comparison to the economic power of the market place. This is particularly the case for small and medium-sized companies, which are probably also more likely than larger companies to be tempted by such tactics. Therefore, if a large body of investors were to disagree with a rating or recommendation, this would soon be reflected in the market value it correspondings to. (Additionally, standard measures against insider trading would apply.)

A first approach could be to create a stock associated with the perception of each service, or each aspect of each service. Thus, one could imagine stocks tracking the perception of the *quality* of a company's products, the perception of the *value* of these, and the perception of the *service* provided. However, stocks have the drawback of being less volatile than other financial instruments, which translates into a lower timeliness of a recomendation mechanism built of stocks. Also, the lower profits achievable by stocks may make the recommendation less accurate even in a (hypothetical) stable state of the system. Finally, stocks offer less flexibility than some other instruments – we will see examples of the usefulness of such flexibility onwards in our description.

In order to achieve increased volatility, we propose the use of *futures* and *futures options*. Unlike common futures and futures options, ours would relate not to the *expected price of a commodity* but to the *perceived quality or value of a service*. While either futures or futures options may be used for our mechanism, and both may be employed at the same time, the user interfaces will differ for the two. We will discuss this in detail after having presented the workings of the recommendation mechanism.

The market values of the futures (resp. futures options) indicate the perception of the associated services or companies. The relationship between the prices associated with two competing companies will, similarly, specify a ranking of the companies. We will show that financial arbitrage will automatically cause a linear ordering of all of the companies being compared. Such an ordering may be performed with respect to each aspect (such as quality, value, service) that corresponds to a future or futures option.

4 Overview of Relevant Financial Instruments

Futures: Long and Short. Traditionally, a *futures contract* is a promise to buy or sell a certain quantity of goods at a given time. To be *long* means to have agreed to *obtain delivery* at the delivery month of the contract, while being *short* means to have agreed to *make delivery* according to the contract. Either way, it is the case that the price of the delivery is agreed upon at the time the position is taken. Futures were introduced as a type of insurance: If, in May, a farmer takes a *short* position for delivery of wheat in September (corresponding to the quantity of wheat he anticipates obtaining at harvest), then he is able to guarantee a profit for his wheat corresponding the contractual price at the time of taking the postion. Similarly, a wheat consumer (such as a baker, perhaps) may take a *long* position to avoid that price fluctuations of wheat alter his calculated profits.

Speculation. It is not necessary to either *have* or *want* wheat in order to buy wheat futures. If a trader believes that the wheat price is about to fall, he will go *short*, at which time he promises to deliver at the price specified in the contract. Later, he could either buy the wheat and make delivery, or more commonly, go *long* to cancel out his previous position. If the price fell during this time, the delivery price he is offered at the early point in time is going to be higher than the price he has to pay to *avoid* making delivery at the latter point in time. The trader will thus make a profit. Of course, this goes both ways, and if the price were to go up, then our trader will accrue a corresponding loss. Similarly, an investor who believes that the price of some merchandise will go up would take a long position at first, and later cancel his position by going short with the same quantity.

Arbitrage. Assume that the exchange rate of pounds to dollars is 1.3; that the rate of dollars to marks is 1.2; and that the rate of marks to pounds is 1.1. Clearly, this is unsustainable, since an investor could start with a small amount of pounds; exchange those for dollars, the dollars for marks, and the marks for pounds; after which he would end up with more pounds than he started with. This process, called arbitrage, is what will immediately impose a linear ordering of the currencies by applying increased upward or downward pressure on the value of at least one of them.

Open Interest. For each long position created, one short position is also created. If a person holding one type takes a position of the opposite type, we say that the two posistions *cancel.* The *open interest* is a count of the number of non-cancelled positions held for each type of future. As such, the open interest indicates the activity of the market; the trading volume is another such measure.

Earnest money. We mentioned that the payment for the commodity is performed at delivery. However, at the time the postition is taken, both sides of the contract deposit *earnest money,* which is typically a fraction of the contractual price. If an investor takes a long position and the prices go up, then he will be able to *withdraw* against his earnest money (since less is needed with the new rate). The same holds for a short position and a falling price. On the other hand, a long investor would under falling prices have to *deposit more* earnest money to keep a security margin. If this margin is ever reached, the clearinghouse would have to limit the number of positions held by the investor, i.e., sell some of the positions at market prices.

Spreads. A *spread* is one long position and one short position – for two different but related types of commodities. The spread is a useful tool for the investor who believes that he knows how the prices of the two types of merchandise will develop *relative to each other* – but without wanting to make bets on how their individual values develop over time. As an example, an investor who takes a position *long Deutsche Mark / short Swiss Franc* believes that the former currency will gain in relation to the latter. As long as this happens, the investor will make a profit, independently of whether they both should go up or both should fall. A spread will typically require less earnest money than a single futures position, as the losses of one side will be balanced to a large extent by the profits of the other.

Futures Options. A *futures option* is a contract that gives the *possibility* of purchasing (resp. selling) a quantity of a commodity at a price specified in the contract. However, it is – unlike normal futures – not *forcing* the buyer of the option to do so. The delivery date of a futures option represents the last point in time when a buyer may exercise the option. The price of the futures option is related to the anticipated price developments of the underlying commodity. Thus, in a bullish market, the price of the *long* futures option is going to be high, and the price of the corresponding *short* position low. (This is similar to how the odds of the favorite race horse will be better than those of a relative newcomer, and the potential payoff the opposite.) It is possible to require the seller of a long (resp. short) futures option to be in possession of the corresponding long (resp. short) future to limit the amount of earnest money demanded by him.

5 Building a Recommendation Mechanism

Delivery. Delivery rarely takes place in a market where a large portion of the investors are speculators. In our setting, this will be even more pronounced, since

for "perception futures", there will be no commodity to be delivered. Therefore, the investors *always* have to cancel their positions at or before the contractual delivery date. It is possible to imagine a futures contract with an *infinite delivery date*. In a futures system with infinite delivery dates, investors would never be forced to cancel out positions, and money would be made by withdrawing against the earnest money, or by voluntarily cancelling out positions. Similarly, an investor holding a losing position has the choice of depositing more earnest money, or to close out the position and get some portion of the deposited earnest money back. If he does not deposit more earnest money to an investment with continously falling value, the clearinghouse will cancel out the position before the earnest money is depleted.

Choice of delivery dates. We propose the use of infinite delivery dates to track behavior of a non-seasonal type, such as the hit ratio of browsers, or the services offered by film developing companies. On the other hand, it may be beneficial to retain normal delivery dates for services of seasonal or periodic nature, such as the value of a tourist resort with different seasonal activities offered. For simplicity, we will focus on futures with infinite delivery dates, as these seem to be more useful for smooth tracking and recommendations for the applications we have in mind. On the other hand, we only consider futures options with normal delivery dates, since this simplifies the risk analysis for the seller of the option, and therefore increases trading volume. We note that it is possible to combine the use of futures having infinite delivery dates with futures options having normal delivery dates (although sellers of the options will find it harder to hedge properly.)

Buying individual futures. If a user (or a corporate investor) believes that a certain service is improving, then he will take a long position for the service. Should he be right, then other investors will follow, and the price will increase, giving the investor a profit. (Unless it is common knowledge that the service is improving, in which case it is to some extent already factored into the price.) Similarly, if the investor believes that a service is becoming worse, he will sell it short.

Using multiple perspectives. It is possible to have two or more sets of futures describing one set of company or services, but from different demographic perpectives. One of these perspectives, for example, can be "as perceived by black and latino men between the ages of 20 and 25" while another may be "as perceived by white teenage women." We note the direct application of the corresponding rankings not only for recommendation systems, but also for purposes of directed advertisments, and research on demographics and consumer behavior.

Using spreads. If an investor believes that company or service A is better than company or service B, he can create a spread position by buying A long and B short. (We note that this fuels the market price of A and cools the market price

of B.) Of course, the decision has to be made after studying market values: if A trades at a much higher value than B, then it is commonly known that their service is better. However, an investor who can identify a situation where they are similarly priced, or even, where B is priced higher, would be likely to want to take the above position. If two different futures describes the same company or service, but from different points of view, then it is possible to create a spread position between these two futures, corresponding to making a bet on what consumer group the service will advance the most onwards.

A note on price movements. As is normal in a free market, downward pressure on the price of a commodity will cause its price to drop. Similarly, upward pressure will result in prices going up. Therefore, the combined effect of investor purchases and sales will move the price of the commodity – in our case the future or futures option – to the level where, according to the market, it belongs.

Translating prices into recommendations. For each set of futures in the same type of market, it will be possible to rank the corresponding companies or services according to the price of the futures, giving the highest ranking to the company or service with the highest market price, etc. Here, the *same type of market* is used to mean when the corresponding services or companies can be *compared.* For example, one can compare the value of products of companies in the same business in a meaningful way, but one cannot compare the value of products between companies in entirely different businesses. Similarly, one cannot compare the value of products offered by one company to the delivery speed of a second company – whether they are in the same business or not. The clearinghouse, or any observer of the market, can create rankings of companies and services that can be compared. Each company or service can be ranked with respect to one or more aspects.

Average Rankings. If there are multiple futures describing one and the same service, but from different perspectives, one can clearly use this for recommendations geared towards the various consumer groups. One may also create *average rankings* by generating a weighted average (where the weights may be selected in proportion to the open interest of the individual futures) of the prices, which then would be translated into a ranking.

Determining the precision of a recommendation. The open interest is an indicator of the number of long and short positions held at the time. A large open interest, combined with a high volume of transactions, is an indicator of a high public interest in the corresponding future. This, in turn, translates to a high degree of precision of the ranking derived from the market prices. On the other hand, a large open interest without any noticeable trading suggests that the price may be about to move, but that the losing side of the trend is not yet convinced of the direction of the movements. If the open interest is very low, and transaction volume limited, then the precision one can obtain is low, as the

opinion is based on only a few investors. Finally, if the open interest is low, but the transaction volume is high, then the trend can be seen as an indication of reasonable precision. (Thus, trends can be seen as a tie-breaking aspect used in the ranking, which primarily is based on the market prices.) In all of the above, large vs. small open interest must be seen as a fraction of the market of options related to the option in question; similarly, the trading volume must be seen in the perspective of total trading in the related market.

Interpreting and using futures options. If futures options are used in combination with futures, one can base the recommendation mechanism solely on the market prices of the futures, and allow the futures options merely to be another tool for trading and putting upwards and downwards pressure on the market values. Furthermore, one can use the discrepancy in the short and long prices for fututes options to determine the trend, i.e., whether there is upwards or downwards pressure on the price of a future. The probably biggest benefit of futures options in our setting is that they do not require constant monitoring of the earnest money, but rather, once a position is taken, the investor may detach himself from further involvement until he decides to exercise the option (i.e., collect the profit, if any.) This makes them particularly practical for "casual investors".

The effects of support purchases. If a company attempts to improve its image by means of performing support purchases of its own futures, then this will cause the value of competing futures to rise by means of arbitrage and new spread positions taken by investors who notice the discrepancy in futures prices and quality of service. In order to sustain the improved rating, the company therefore has to keep making support purchases to counter the market forces. We will study the cost of this in section 7. We note that a company can also improve its image by attempting to damage the image of its competitors. However, this results in a larger cost (to get the same relative improvment in the rating) as long as the cumulative market of the competitor's futures has larger volume than the company's own futures alone. Therefore, we will focus on the former threat.

6 User Interface

Displaying recommendations. According to the methods described above, one can create rankings of the various aspects of a service or company of interest to a user. These can be used for decision guides by allowing the user to prioritize the various aspects, and weighing the rankings obtained by the user-defined weights. One can also display the available services on a line, their positions set as a function of their traded value, with arrows indicating trends. Additionally, one may add a measure of the precision, using, e.g., a pie chart under each company on the axis. (Here, the precision, as mentioned, can be estimated from the open interest and the trading volume, as well as the current prices of its futures options.)

Creating an account. Everybody who wants to start investing in *perception futures* or *perception futures options* needs to transfer money to a clearinghouse or a broker. A variety of well understood methods can be used for this, as well as for protecting the account against unauthorized access. Once an account has been created and a minimum balance established, the user can trade in available options and futures.

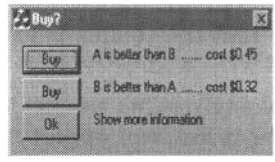


Fig. 1. Example interface.

Displaying investment information. Casual investors may use an interface as that depicted in figure 1. The investor may request more details, and can assess the approximate and relative market values of futures by looking at the related recommendation display. By selecting one or more companies, he can then obtain the prices for futures and futures options for each one of these, along with the prices associated with creating spreads for the various combinations. Delivery dates – when applicable – would be displayed as well. The user would also be given access to information on his existing positions, and be given the possibility of closing out or strengthening these. The more thorough investor would also be given information that allows him to assess the open interest, trading volume, and prices as functions of time.

7 Analysis

The following analysis based on a set of simplified assumptions meant to examplify how to perform economical reductions. We do not attempt to derive the correct models (and we are aware of the likely complexity of these), and therefore advise the reader to take the conclusions of the analysis with a big grain of salt and merely think of these as indicative of the economic reductions possible, once market data is available. Having said that, let us now consider how an economic reduction can be made, given a model of the market behavior:

Let us fix a company or service to be analysed. We let x correspond to the number of clients it has per time unit, and y the total number of clients per time unit in its niche. We will let δ be the average perceived over-valuation of its future by consumers and investors comparing it to other companies. We will finally let ϵ be the actual amount of over-valuation of its future caused by investment of the company itself. Both δ and ϵ are assumed to be fractions larger than one.

However, a similar argument will hold for depreciation of competitors' futures price.

Assumption 1. We assume that the number of *eyeballs* a future receives per time unit is $E = c_1 x + c_2 y$, where c_1 and c_2 are constants.

Assumption 2. We assume that the trading volume of a future on average is proportional both to its perceived over-valuation δ and to the number of eyeballs E. In particular, we assume that the volume is $V = \delta E$.

Assumption 3. For ease of analysis, we assume that $\delta = \nu\epsilon$, meaning that the average investor perceives the investment opportunity as a fraction ν of the real cost discrepancy.

Assumption 4. We assume that the market is rational in that the common investor tries to maximize his financial benefit.

Loss per time unit. The cost of improving the company's ranking by support-buying of its futures is the cost of the volume of futures traded by the cheater. Thus, the cheater's finacial loss per time unit is $L = \epsilon V = \epsilon\delta E$ (wherein we rely on assumption 2.) Using assumption 3, we can simplify this to $L = \nu\epsilon^2 E$, which, according to assumption 1 is $L = \nu\epsilon^2(c_1 x + c_2 y)$. Consider now the ratio of the cost L taken per customer x. This is the *average support spending per customer* (in a steady state.) We plot the ratio L/x as a function of the discrepancy ϵ in figure2, using $(c_1, c_2, \nu) = (1/100, 1/500, 0.9)$ as possible values and market shares (i.e., ratios x/y) between 1/100 and 1/2.

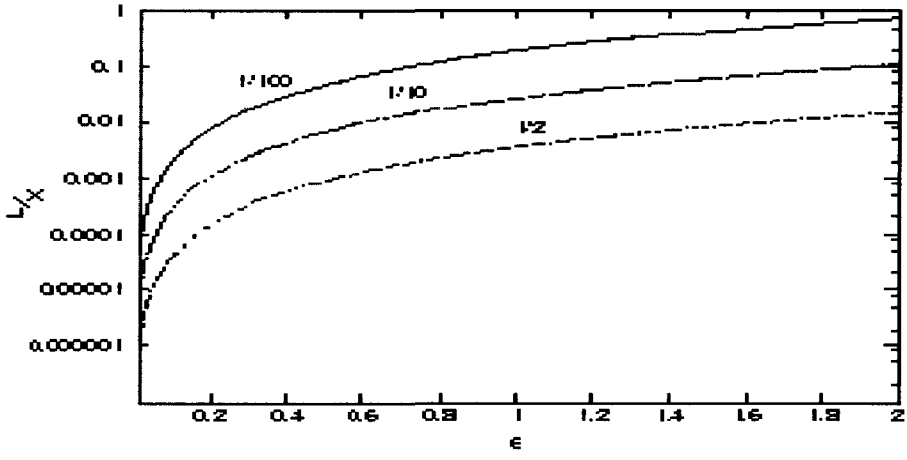


Fig. 2. The cost as a function of the degree of manipulation.

We see that – under our assumptions, and using our system – the losses increase rapidly with the distortion of the futures value. We also see that the smaller the market share, the larger the overhead of cheating. This is very convenient, as larger companies are more likely to be carefully audited, and therefore less likely to even attempt purchasing their image on the futures market.

We note that the costs must be put in relation to the anticipated profits per time unit stemming from the improved rating. However, one must also consider alternative ways of obtaining this better rating, whether to advertise or to improve the products or services. The rating system is abuse-free if these latter costs are lower than those of support-buying futures.

8 Conclusions

We have presented a novel mechanism for producing recommendations. Our mechanism is light-weight, has resistance against manipulation, and is timely. It will not need any new software to be distributed to the average user (who may use a standard browser), and it will be easy to use.

Acknowledgements. Many thanks to Matt Franklin for inspiration and fruitful discussions, and to the anonymous referees for improving the presentation.

References

1. Better Business Bureau, `www.bbb.org`
2. D. Conley, "A free market election failure," `www.salon.com/tech/feature/2000/11/16/election_prediction/index.html`
3. eBay, `www.eBay.com`
4. Iowa Electronic Markets, `www.biz.uiowa.edu/iem/index.html`
5. Lonely Planet, `www.lonelyplanet.com`
6. Cartoon in the New Yorker, p. 61, July 5 '93.
7. Zagat reviews, `www.zagat.com`

Secure Combinatorial Auctions by Dynamic Programming with Polynomial Secret Sharing

Koutarou Suzuki[1] and Makoto Yokoo[2]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation
1-1 Hikari-no-oka, Yokosuka, Kanagawa, 239-0847 Japan
http://info.isl.ntt.co.jp/~koutarou/
koutarou@isl.ntt.co.jp
[2] NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
http://www.kecl.ntt.co.jp/csl/ccrg/members/yokoo/
yokoo@cslab.kecl.ntt.co.jp

Abstract. Combinatorial auctions have recently attracted the interests of many researchers due to their promising applications such as the spectrum auctions recently held by the FCC. In a combinatorial auction, multiple items with interdependent values are sold simultaneously and bidders are allowed to bid on any combination of items. This paper presents a method for implementing several secure combinatorial auction protocols based on our newly developed secure dynamic programming protocol. Dynamic programming is a very effective, widely used technique for tackling various combinatorial optimization problems, including several types of combinatorial auctions. Our secure dynamic programming protocol utilizes secret sharing techniques and can obtain the optimal solution of a combinatorial optimization problem, i.e., result of a combinatorial auction, without revealing the inputs of the problem, i.e., bidding prices. We discuss the application of the method to several combinatorial auctions, i.e., multiple-unit single-item auctions, linear-goods auctions, and general combinatorial auctions.

Keywords: dynamic programming, combinatorial auction, spectrum auction, secret sharing

1 Introduction

Internet auctions have become an especially popular part of Electronic Commerce. Some studies on Internet auctions have already been conducted [18,34], and combinatorial auctions have recently attracted considerable attention [8,13, 14,15,26,27,36]. In contrast with conventional auctions that sell a single item at a time, combinatorial auctions sell multiple items with interdependent values simultaneously and allow the bidders to bid on any combination of items.

In a combinatorial auction, a bidder can express complementary/substitutional preferences over multiple bids. For example, in the Federal Communications Commission (FCC) spectrum auction [17], a bidder

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 44–56, 2003.

may desire licenses covering adjoining regions simultaneously (i.e., these licenses are complementary), while being indifferent as to which particular channel was awarded (channels are substitutional). By considering the complementary/substitutional preferences, we can increase the participants' utility and the revenue of the seller. To execute a combinatorial auction, we need to solve a combinatorial optimization problem called the winner-determination problem, i.e., we need to find the combination of bids with disjoint sets of goods, such that the sum of the bidding prices is maximized. The winner determination problem has been tackled using various optimization techniques recently [8,22, 25,27].

On the other hand, from the view point of security, to hide bidding prices is an important problem, and there are many researches on secure auction protocols [1,2,5,6,7,10,11,12,19,20,23,24,29,30].

If we can trust the auctioneer, we simply gather all private information (i.e., bidding prices) at the auctioneer, who can then solve the problem using any available centralized optimization technique. However, we cannot take it for granted that there exists such a trusted auctioneer. For example, in a standard first-price sealed-bid auction [21], where the highest bidder wins and pays his/her own price, the auctioneer might collude with a particular participant and reveal information of incoming bids to that participant during the auction.

If we use a strategy-proof mechanism, such as a second-price sealed bid (Vickrey) auction [21], where the highest bidder wins and pays the second highest price, the information of other participants' bids becomes useless; thus we can discourage such collusion between the auctioneer and bidders. However, in a second-price sealed-bid auction, if the auctioneer can know the highest bid, he/she can increase his/her revenue by fabricating a fake bid whose price is very close to the highest bid.

We can utilize various cryptographic technologies to ensure that the auctioneer cannot learn bidding prices while accepting incoming bids. However, if the auctioneer can know bidding prices even after the auction ends, he/she can utilize the information of the bids for future auctions. For example, the auctioneer learns the behavior/preference of a certain participant from past auctions and conducts fraudulent activities based on this information, or the auctioneer might reveals/sells such private information to others. Varian [33] pointed out this problem as follows: *"Even if current information can be safeguarded, record of past behavior can be extremely valuable, since historical data can be used to estimate the willingness to pay. What should be the appropriated technological and social safeguards to deal with this problem?"*.

This paper aims to provide a solution to this problem: in a combinatorial auction, multiple auction servers can cooperatively solve the winner determination problem, i.e., they can find the combination of bids that maximizes the sum of the bidding prices, while the information of bids that are not part of the optimal solution is kept secret even from the auction servers. More specifically, by utilizing secret sharing techniques [28], we develop a method for securely performing a dynamic programming algorithm [3] that is very effective and widely

used against various combinatorial optimization problems. We show also how to use our secure dynamic programming protocol in various types of combinatorial auctions.

The method proposed in this paper is based on the method proposed for $M+1$-st price auctions by Kikuchi [11]. His method represents the bidding price as the degree of a polynomial, and utilize the fact that the maximum of the degrees of two polynomials can be obtained from the degree of the sum of the two polynomials. In this paper, to implement a secure dynamic programming protocol, we additionally utilize the fact that the sum of the degrees of two polynomials can be obtained from the degree of the product of the two polynomials.

The rest of this paper is organized as follows. In Section 2, we briefly review dynamic programming techniques. In Section 3, we present our newly developed secure dynamic programming protocol. In Section 4, we describe how to apply the proposed method to various types of combinatorial auctions. In Section 5, we discuss its relation to existing techniques.

2 Dynamic Programming

Dynamic programming [3] was developed by R. Bellman during the late 1950's. Dynamic programming is a powerful method that can be applied to various combinatorial optimization problems.

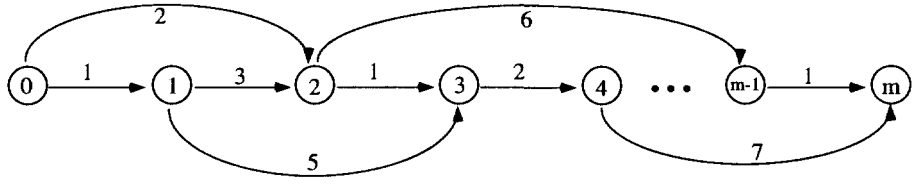


Fig. 1. Example of one-dimension directed graph

In the following, we use the problem of finding the longest path in the one-dimensional directed graph described in Figure 1 to illustrate the concept of dynamic programming. This graph consists of nodes $0, 1, 2, \ldots, m$ with directed links among them. A link is represented as (j, k), where $j < k$. For each link (j, k), the weight $w(j, k)$ of the link is given. The goal is to find the longest path from initial node 0 to terminal node m, i.e., to find a path from 0 to m such that the sum of the weights of the links is maximized. For simplicity, we assume that for each node j (where $0 \leq j < m$), there exists at least one link that starts from j, i.e., there is no dead-end node except m.

One notable characteristic of this problem is as follows. Assume L is the longest path from 0 to m. It follows that for any node j on L, the last half of L, i.e., the part of L from j to m, is also a longest path from j to m. This characteristic is called the *principle of optimality*. This feature enables us to

find the optimal solution of the original problem from the optimal solutions of sub-problems.

More specifically, we can obtain the length of the longest path from 0 to m by solving the following recurrence formula from node $m-1$ to 0. In this formula, $f(j)$ represents the length of the longest path from j to m. We call $f(j)$ the *evaluation value* of node j. For terminal node m, $f(m)$ is defined as 0. For initial node 0, $f(0)$ represents the optimal solution, i.e., the length of the longest path from 0 to m.

$$f(j) = \max_{(j,k)} \{w(j,k) + f(k)\}$$

When calculating this formula, for each node j, we record the link (j,k) that gives the evaluation value $f(j)$, i.e., the link that gives $\max_{(j,k)}\{w(j,k)+f(k)\}$. We can construct the longest path by following these recorded links from 0 to m.

We give below a generalization that is used in Section 4. There are $m+1$ stages $j = 0, 1, \ldots, m$ and states $\{(j,s)\}$ at each stage j. There can be directed links $((j,s),(k,t))$ between these states only if $j < k$, i.e., there are no links at the same stage nor from higher stage to lower stage. For each link, weight $w(((j,s),(k,t)))$ is given. Dynamic programming evaluates function f defined by the following recurrence relation

$$f((j,s)) = \max_{j<k,\ ((j,s),(k,t)):\text{link}} \{w(((j,s),(k,t))) + f((k,t))\}.$$

We can compute value $f((0,s))$, that is the optimal value of the original problem, by iterative application of the relation for $j = m, m-1, \ldots, 0$ with initial values $f((m,s)) = iv(s)$.

In the rest of this paper, we describe our secure dynamic programming protocol based on the longest path finding problem in a one-dimensional directed graph. As discussed in Section 4, the application of the proposed protocol to the general case is also straightforward.

3 Secure Dynamic Programming

The proposed secure dynamic programming protocol is presented below together with a discussion of its security and efficiency.

3.1 Requirements

The requirements for our secure dynamic programming protocol is as follows:

- Each weight publisher sends information of the weight of one link to evaluators.
- Evaluators cooperatively execute dynamic programming and find the optimal solution, while each weight is kept secret.

To realize this protocol, we have to answer the following question: how can we determine the maximum and sum of weights without revealing the weights themselves? The decision on how to represent and encrypt the weight is crucial to making these tasks feasible. Our approach is to represent a weight as the degree of a polynomial; thus the maximum/sum of the degree of two polynomials can be obtained by the degree of the sum/product of the two polynomials.

3.2 Preliminaries

We start by explaining the secret sharing used in our protocol. This secret sharing is based on Shamir's polynomial secret sharing [28], but differs from it at the point that we represent a secret by the degree of a polynomial while Shamir uses the constant term of a polynomial. This kind of polynomial secret sharing is also used for $M+1$-st price auctions by Kikuchi [11].

Weight publisher P who has a secret $s \in \mathbf{Z}_p$ performs secret sharing as follows. Weight publisher P randomly chooses n $(n > s)$ points $x_1, x_2, \ldots, x_n \in \mathbf{Z}_p$ and constant $c \in \mathbf{Z}_p$ and publishes them, and randomly chooses polynomial $A \in \mathbf{Z}_p[x]$ s.t. $deg(A) = s$ and $A(0) = c$ and holds it secret. Weight publisher P then sends l-th share $A(x_l)$ to l-th evaluator E_l through a secure channel.

In this situation, we can examine whether $deg(A) \leq d$ or not while polynomial A is kept secret. First, mask publisher T randomly chooses mask polynomial $M \in \mathbf{Z}_p[x]$ s.t. $deg(M) = d$ and $M(0) = 0$ and keeps it secret, and sends l-th share $M(x_l)$ to l-th evaluator E_l through a secure channel. Next, $d+1$ evaluators $E_1, E_2, \ldots, E_{d+1}$ publish masked shares $A(x_l)+M(x_l)(l = 1, 2, \ldots, d+1)$. Using these $d+1$ masked shares, we perform polynomial interpolation, i.e., determine polynomial $A+M$, recover $A(0) = A(0)+M(0)$, and check whether $A(0) = c$ or not. If $deg(A) \leq d$, we have $deg(A+M) = d$ and can recover the constant term $A(0) = c$ from $d+1$ shares. If $deg(A) > d$, we have $deg(A+M) > d$ and cannot recover the constant term $A(0) = c$ from $d+1$ shares. Hence, if $A(0) = c$ holds, we are convinced that $deg(A) \leq d$

Furthermore, the maximum/sum of the degree of two polynomials can be obtained by the degree of the sum/product of the two polynomials by the following formulae

$$max\{deg(A), deg(B)\} = deg(A+B)\ ^{\star},\ deg(A) + deg(B) = deg(A \cdot B),$$

respectively. Each evaluator E_l can compute, locally, its share of sum $A+B$ / product $A \cdot B$ of two polynomials A and B by taking sum $A(x_l)+B(x_l)$ / product $A(x_l) \cdot B(x_l)$ of two shares $A(x_l)$ and $B(x_l)$. This allows the maximum/sum of two secrets to be locally determined.

---

* The highest terms may be canceled out with probability $\frac{1}{p}$. We can ignore this case since the probability is negligible.

3.3 Secure Dynamic Programming

There is weight publisher $P_{(i,j)}$ for link (i,j), plural e evaluators $E_1, \ldots, E_e$ where e is greater than the length of the longest path, and $t_m + 1$ mask publishers $T_0, \ldots, T_{t_m}$ where t_m is a threshold parameter of mask publishers. In the following Step 3 and 4, each mask publisher T_l publishes a part of mask M_l, and sum $M = M_0 + \ldots + M_{t_m}$ is used as the mask. Our protocol consists of the following steps:

- Step 1 : Weight publisher $P_{(i,j)}$ of link (i,j) sends to evaluator E_l l-th share of the weight of link (i,j).
- Step 2 : Each evaluator E_l computes, locally, the recurrence relation of dynamic programming, to obtain the l-th share of the optimal value.
- Step 3 : Evaluators obtain the optimal value cooperatively.
- Step 4 : Evaluators trace the links back to obtain the optimal path.

First, mask publisher T_0 randomly chooses e points $x_1, x_2, \ldots, x_e \in \mathbf{Z}_p$ and constant $c \in \mathbf{Z}_p$ and publishes them.

In Step 1, Weight publisher $P_{(i,j)}$ decides weight $\tilde{w}(i,j)$, extends weight $w(i,j) = \tilde{w}(i,j) + t_w \times (j-i)$ where t_w is a threshold parameter of weight publishers. By this extension, the optimal solution, i.e., the longest path from node 0 to m will not change. We denote by $\tilde{f}(i)$ / $f(i)$ the evaluation value of node i using the original weights $\tilde{w}$ / the extended weights w. Then, $f(i) = \tilde{f}(i) + t_w \times (m-i)$ holds for each node i. Thus we can find the maximum and perform secure dynamic programming in the same manner as described in the preliminaries with slight modification. Weight publisher $P_{(i,j)}$ then randomly chooses polynomial $W_{(i,j)} \in \mathbf{Z}_p[x]$ s.t. $deg(W_{(i,j)}) = w(i,j)$ and $W_{(i,j)}(0) = c$ and holds it secret. Weight publisher $P_{(i,j)}$ then sends l-th share $W_{(i,j)}(x_l)$ to l-th evaluator E_l through a secure channel.

In Step 2, Each evaluator E_l computes, locally, the following formula

$$F_j(x_l) = \sum_{(j,k)} W_{(j,k)}(x_l) \cdot F_k(x_l)$$

for $j = m-1, m-2, \ldots, 0$ with $F_m(x_l) = 1$. This formula corresponds to the recurrence relation of dynamic programming

$$f(j) = \max_{(j,k)} \{w(j,k) + f(k)\},$$

thus we have $deg(F_j) = f(j)$.

In Step 3, As described in the preliminaries, we can check whether $deg(F_0) \leq d$ or not. (Notice that we have to check whether $F_0(0)$ is equal to the appropriate constant determined by constant c and the one-dimensional graph or not. For example, if we set $c = 0$, $F_0(0)$ should be 0.) By using this check, we can perform binary search, find the optimal value $f(0) = deg(F_0)$, and publish it.

In Step 4, we obtain the optimal path that attains the optimal value $f(0)$ as follows. Consider that we know $f(j) = deg(F_j)$ and want to find node k s.t.

$deg(F_j) = deg(W_{(j,k)} \cdot F_k)$. We examine whether $deg(W_{(j,k)} \cdot F_k) \leq deg(F_j) - 1$ or not for all nodes k linked to node j. Since the inequality holds for node k that does not attains $f(j)$, we are convinced that the node k attains $f(j)$ if the inequality does not hold for node k. After we find the node k that attains $f(j)$, we determine $f(k) = deg(F_k)$ as in Step 3, and publish it. Iterating these process recursively yields the optimal path.

3.4 Example

We give an example for the one-dimension graph shown in Figure 2.

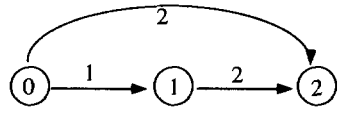


Fig. 2. One-dimension graph

There are three links $(0,1)$, $(1,2)$, $(0,2)$ with weights 1, 2, 2, respectively. Weight publishers $P_{(0,1)}$, $P_{(1,2)}$, $P_{(0,2)}$ for these links generate the following polynomials

$$W_{(0,1)} = x,\ W_{(1,2)} = x^2 - x,\ W_{(0,2)} = 2x^2 + 2x.$$

There are four evaluators E_1, E_2, E_3, E_4, with $x_1 = -2$, $x_2 = -1$, $x_3 = 1$, $x_4 = 2$. For simplicity, we set $t_w = 0$ and $t_m = 1$. We also assume $c = 0$.

Table 1. Shares

	$W_{(0,1)}$	$W_{(1,2)}$	$W_{(0,2)}$	F_1	F_0	M	$W_{(0,1)} \cdot F_1 + M$	$W_{(0,2)} \cdot F_2 + M$
E_1	-2	6	4	6	-8	8	-4	12
E_2	-1	2	0	2	-2	2	0	2
E_3	1	0	4	0	4	2	2	6
E_4	2	2	12	2	16	8	12	20

Table 1 shows the shares of $W_{(0,1)}$, $W_{(1,2)}$, $W_{(0,2)}$, and the shares of F_1 and F_0 as computed by each evaluator in Step 2. From shares $F_0(-2) = -8$, $F_0(-1) = -2$, $F_0(1) = 4$, $F_0(2) = 16$, we can recover $F_0(x) = x^3 + x^2 + 2x$ of degree 3 with constant term $F_0(0) = 0$. This convinces us that $f(0) = 3$.

Table 1 also shows masked shares with $M = 2x^2$ in Step 4. Since the constant term of the polynomial of degree 2 recovered by shares of $W_{(0,1)} \cdot F_1 + M$ is not equal to 0, we are convinced that link $(0,1)$ attains $f(0) = 3$.

3.5 Security

In this section, we discuss the security of our protocol against the passive adversary that can see public information and all information of collusive participants, and tries to find secret information; it cannot manipulate the collusive participants.

Notice that a value at a point of a polynomial that is uniformly randomly chosen is uniformly random. In the protocol, all published shares are masked with random polynomials generated by $t_m + 1$ mask publishers. Hence, if the number of collusive mask publishers is less than or equal to the threshold t_m, masked shares are uniformly random to the adversary and leak no information.

By the extension of weight, extended weight $w(i,j) = deg(W_{(i,j)})$ is more than or equal to t_w. Hence, if the number of collusive weight publishers is less than or equal to threshold t_w, the adversary cannot recover polynomial $W_{(i,j)}$ and cannot obtain any information about the weight.

Thus, our protocol is unconditionally secure against the passive adversary.

Theorem 31 *The proposed protocol unconditionally hides all information except the optimal value $f(m)$ and optimal path against a passive adversary with t_m or less collusive mask publishers and t_w or less collusive weight publishers.*

Finally, we mention that each weight publisher can publish dummy weights for all links at the lowest value to conceal for which link he publish his weight. In the auction scenario, this means that each bidder can cast dummy bids for all goods at the lowest price to conceal which goods he want to buy.

3.6 Efficiency

Table 2 shows communication pattern, round complexity, and volume per roud through secure channels of each step. We omit communications without secure channels, i.e., evaluators publish shares in step 3 and 4, which can be implemented by a bulletin board. Here, l is the number of links, n is the number of nodes, e is the number of evaluators (which is equal to or greater than possible maximal value), $t_m + 1$ is the number of mask publishers, and p is the order of the finite field $\mathbf{Z}_p$.

The round complexity is proportional to the number of links or nodes, so our protocol can handle certain kinds of combinatorial auctions with large number of items. However, the round complexity is also proportional to the number of evaluators, so complexity becomes large for wide price ranges.

Table 2. Communication complexity

	pattern	round	volume
Step 1	$P_{(i,j)} \to E_k$	$l \times e$	$\log p$
Step 2	—	0	0
Step 3	$T_i \to E_k$	$(t_m + 1) \times e \times \log e$	$\log p$
Step 4	$T_i \to E_k$	$(t_m + 1) \times e \times (l + \log e \times n)$	$\log p$

4 Application to Combinatorial Auctions

In this section, we discuss the application of our secure dynamic programming protocol to several types of combinatorial auctions, i.e., multi-unit auctions, linear-goods auctions, and general combinatorial auctions.

4.1 Multi-unit Auctions

In multi-unit auctions, m units of an identical item are auctioned. Each bidder B_k $(1 \leq k \leq N)$ declares his/her bidding price $b_k(j)$ for each quantity j, where $0 \leq j \leq m$. The goal is to find the allocation that maximizes the sum of the bidding prices.

The optimal allocation in a multi-unit auction can be obtained by solving the following recurrence formula [31,32].

$$f((1,j)) = b_1(j), \quad f((k,s)) = \max_{0 \leq j \leq s} \{f((k-1, s-j)) + b_k(j)\}.$$

In this formula, $f((k,s))$ represents the value of the optimal solution of a subproblem, i.e., optimally allocating s units among k participants, i.e., bidders $B_1, B_2, \ldots, B_k$. The optimal solution is given by $f((N,m))$.

Applying our secure dynamic programming protocol to this problem is rather straightforward.

4.2 Linear-Goods Auctions

In a linear-goods auction [22,31], there exist m goods $G = \{1, 2, \ldots, m\}$. These goods are sequentially ordered. Each bidder $B_k (1 \leq k \leq N)$ bids his/her bidding price $b_k([l_k, u_k])$ for an interval $[l_k, u_k] \subseteq G$ of the goods, i.e., a bidder wants to obtain a continuous sequence of goods. We assume each bidder bids for a single interval (or equivalently bids for several intervals independently). The result of the auction is the allocation of the m goods that maximizes the sum of all bidders' bidding prices.

Auctions for linear-goods can be used for time scheduling (e.g., for the allocation of time slots in a conference room), or for the allocation of a one-dimensional space (e.g., for parts of a seashore), or spectrum right auctions in which each bidder (wireless carrier) wants a continuous frequencies to minimize interference between carriers.

This problem can be directly mapped into the problem of finding the longest path in a one-dimensional graph [16]. We consider the graph with nodes $0, 1, 2, \ldots, m$, i.e., there exists an initial node 0 and nodes that correspond to goods. Between these nodes, we place links $(l_k - 1, u_k)(1 \leq k \leq N)$ corresponding to bids $b_k([l_k, u_k])(1 \leq k \leq N)$ and dummy links $(j, j+1)(0 \leq j \leq m-1)$ with weights

$$w((l_k - 1, u_k)) = b_k([l_k, u_k]) \ \ (1 \leq k \leq N), \quad w((j-1, j)) = 0 \ \ (1 \leq j \leq m).$$

The allocation of the m goods that maximizes the sum of all bidders' bidding prices corresponds to the longest path from node 0 to node m, and can be securely computed by our secure dynamic programming protocol. Also, our method can handle multiple links between two nodes.

The idea of linear-goods auctions can be generalized to route auctions, in which each bidder bids for a path in a general graph. The result of the auction is the path from the start node to the destination node that maximizes/minimizes the sum of all bidders' bidding prices.

One application of route auctions is transportation task assignment, in which the auctioneer wants to transport his/her cargo from a starting city to a destination city. There are several transportation companies. Each company bids his/her price to carry the cargo for a path (which can be only a part of the total journey), and the auctioneer chooses the combination of paths that minimizes the total cost. This problem can be formalized as finding the shortest path in a graph, and so can be solved using the secure dynamic programming protocol presented in this paper.

4.3 General Combinatorial Auctions

In a general combinatorial auction, there exist multiple different goods $G = \{1, 2, \ldots, m\}$. Each bidder $B_k (1 \leq k \leq N)$ bids his/her price $b_k(S)$ for each subset $S \subseteq G$. The goal is to find the allocation of goods that maximizes the total price.

As described in [22], this problem can be solved by dynamic programming as follows. For each subset S, $b(S)$ represents the highest price of the subset. For simplicity, we assume that each subset has at least one bid (otherwise, we can put a dummy bid with price 0). For each subset $S \subseteq G$, we create a node S. Between these node S, we place N multiple links

$$\{(S, \phi)_k | S \subseteq G, 1 \leq k \leq N\}, \quad \{(S, C)_k | C \subset S \subseteq G, |C| \geq |S|/2, 1 \leq k \leq N\}$$

where ϕ represents the empty set and $|X|$ is the number of elements of X, with weights

$$w((S, \phi)_k) = b_k(S), \quad w((S, C)_k) = b_k(S \setminus C).$$

The optimal allocation is given by the longest path from initial node G to terminal node ϕ. It is clear that this problem can be solved using the secure dynamic programming protocol presented in this paper. Our method can handle multiple links between two nodes. Therefore, we don't need to pre-select the bid with the highest price for each subset.

One obvious disadvantage of this approach is that the number of nodes becomes very large, i.e., 2^m. However, this seems somewhat inevitable if we are to solve this problem using dynamic programming, since the winner determination problem of a general combinatorial auction is NP-complete [22].

5 Discussions

There have been various works on secure auction protocols [1,2,5,6,7,10,11,12, 19,20,23,24,29,30]. However, as far as the authors' know, there has been no other research on secure dynamic programming nor on secure combinatorial auction based on dynamic programming techniques. Here, we discuss some related works.

Kikuchi [11] developed a secure M + 1-st price auction protocol by using secret sharing. However, in his auction, multiple units of an identical goods are auctioned, but each participant is assumed to want only one unit. Our method can be applied to the cases where each participant wants to buy multiple units.

It is well known that any combinatorial circuit can be computed securely using general-purpose multi-party protocols [4,9]. Therefore, if we can construct a combinatorial circuit that implements a dynamic programming algorithm, in principle, such an algorithm can be executed securely. However, to execute such a general-purpose multi-party protocol, for each computation of an AND gate in the circuit, evaluators must communicate with each other. Using such a general purpose multi-party protocol for large-scale dynamic programming applications is not practical due to the required communication costs.

Naor et al. [19] proposed a general method for executing any auction protocol, including combinatorial auction protocols, by using a technique called garbled circuit [35]. This method is efficient and does not require interactive communications among multiple evaluators. However, we need to construct a circuit that implements a dynamic programming algorithm, and after the construction we cannot change the algorithm, i.e., the directed graph of the dynamic programming or the number of bidders. In our protocol, by contrast, we can change dynamically these things during the auction.

6 Conclusion

In this paper, we presented a secure dynamic programming method that utilizes secret sharing techniques. By using this method, multiple servers cooperatively solve a combinatorial optimization problem, while the inputs are kept secret even from the servers. Furthermore, we discussed its application to several combinatorial auctions, i.e., multi-unit auctions, linear-goods auctions, and general combinatorial auctions.

One limitation of the proposed method is that, we need a large number of evaluators, since the number of evaluators must be proportional to the length of the longest path. It is an important and challenging problem to develop a method that requires a smaller number of evaluators [37].

Although dynamic programming is a very powerful, widely-used combinatorial optimization method, applying dynamic programming techniques to general combinatorial auctions is not feasible for large-scale problems, since it requires an exponential number of nodes. We are now investigating how to securely compute more average-case efficient algorithms [8,26,27].

References

1. Masayuki Abe and Koutarou Suzuki. M+1-st price auction using homomorphic encryption. *Proceedings of Public Key Cryptography 2002*, 2002.
2. O. Baudron and J. Stern. Non-interactive private auctions. *Proceedings of Financial Cryptography 2001*, 2001.
3. R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
4. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of 20th ACM Symposium on the Theory of Computing*, pages 1–10, 1988.
5. C. Cachin. Efficient private bidding and auctions with an oblivious third party. *Proceedings of 6th ACM Conference on Computer and Communications Security*, pages 120–127, 1999.
6. K. Chida, K. Kobayashi, and H. Morita. Efficient sealed-bid auctions for massive numbers of bidders with lump comparison. *Proceedings of ISC 2001*, 2001.
7. Matthew K. Franklin and Michael K. Reiter. The design and implementation of a secure auction server. *IEEE Transactions on Software Engineering*, 22(5):302–312, 1986.
8. Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computation complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 548–553, 1999.
9. Oded Goldreich, Silvio Micli, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of 19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.
10. M. Harkavy, J. D. Tygar, and H. Kikuchi. Electronic auctions with private bids. *Proceedings of Third USENIX Workshop on Electronic Commerce*, pages 61–74, 1998.
11. H. Kikuchi. (m+1)st-price auction protocol. *Proceedings of Financial Cryptography 2001*, 2001.
12. H. Kikuchi, M. Harkavy, and J. D. Tygar. Multi-round anonymous auction protocols. *Proceedings of first IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69, 1998.
13. Paul Klemperer. Auction theory: A guide to the literature. *Journal of Economics Surveys*, 13(3):227–286, 1999.
14. Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auction. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)*, pages 96–102, 1999.
15. Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the Second ACM Conference on Electronic Commerce (EC-00)*, pages 66–76, 2000.
16. Tomomi Matsui and Takahiro Watanabe. Sealed bid multi-object auctions with necessary bundles and its application to spectrum auctions. In *Proceedings of the 4th Pacific Rim International Workshop on Multi-agents (PRIMA-2001)*, pages 78–92. Springer-Verlag, 2001. Lecture Notes in Artificial Intelligence 2132.
17. John McMillan. Selling spectrum rights. *Journal of Economics Perspectives*, 8(3):145–162, 1994.
18. Dov Monderer and Moshe Tennenholtz. Optimal auctions revisited. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 32–37, 1998.

19. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)*, 1999.
20. K. Omote and A. Myaji. An anonymous auction protocol with a single non-trusted center using binary trees. *Proceedings of ISW2000*, pages 108–120, 2000.
21. Eric Rasmusen. *Games and Information*. Blackwell, 1994.
22. Michael H. Rothkopf, Aleksandar Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.
23. K. Sako. Universally verifiable auction protocol which hides losing bids. *Proceedings of Public Key Cryptography 2000*, pages 35–39, 2000.
24. K. Sakurai and S. Miyazaki. A bulletin-board based digital auction scheme with bidding down strategy. *Proceedings of 1999 International Workshop on Cryptographic Techniques and E-Commerce*, pages 180–187, 1999.
25. Yuko Sakurai, Makoto Yokoo, and Koji Kamei. An efficient approximate algorithm for winner determination in combinatorial auctions. In *Proceedings of the Second ACM Conference on Electronic Commerce (EC-00)*, pages 30–37, 2000.
26. T. Sandholm, S. Suri, A. Gilpin, and D. Levine. A fast combinatorial algorithm for optimal combinatorial auctions. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 1102–1108, 2001.
27. Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 542–547, 1999.
28. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
29. S. G. Stubblebine and P. F. Syverson. Fair on-line auctions without special trusted parties. *Proceedings of Financial Cryptography 1999*, 1999.
30. Koutarou Suzuki, Kunio Kobayashi, and Hikaru Morita. Efficient sealed-bid auction using hash chain. *Proceedings of International Conference Information Security and Cryptology 2000 (LNCS 2015)*, pages 183–191, 2000.
31. Moshe Tennenholtz. Some tractable combinatorial auctions. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 98–103, 2000.
32. Stan van Hoesel and Rudolf Müller. Optimization in electronic markets: examples in combinatorial auctions. *Netnomics*, 3:23–33, 2001.
33. Hal R. Varian. Economic mechanism design for computerized agents. In *Proceedings of the First Usenix Workshop on Electronic Commerce*, 1995.
34. Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Proceedings of the Second International Conference on Autonomous Agents (Agents-98)*, pages 301–308, 1998.
35. A. C. Yao. How to generate and exchange secrets. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.
36. Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, 130(2):167–181, 2001.
37. Makoto Yokoo and Koutarou Suzuki. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, 2002. (to appear).

A Second-Price Sealed-Bid Auction with Verifiable Discriminant of p_0-th Root*

Kazumasa Omote and Atsuko Miyaji

School of Information Science
Japan Advanced Institute of Science and Technology,
Asahidai 1-1, Tatsunokuchi, Nomi, Ishikawa, 923-1292 Japan
{omote, miyaji}@jaist.ac.jp

Abstract. A second-price sealed-bid auction is that a bidder who offers the highest price gets a good in the second highest price. This style of auction solves the problems of both an English auction and a first-price sealed-bid auction. An electronic first-price sealed-bid auction cannot directly be applied to a second-price sealed-bid auction which keeps the highest bid secret. We propose the verifiable discriminant function of the p_0-th root. Our auction scheme satisfies public verifiability of auction results, and also does not have a single entity who knows the highest bid value even after an auction. Furthermore the bidding cost of our scheme is lower than that of the previous one.

Keywords: Proof of knowledge, Public verifiability, Economics

1 Introduction

1.1 Background

A *sealed-bid auction* is that each bidder secretly submits a bid to auction manager (AM) only once for an auction. Compared with English auction, a winner is decided more efficiently. In a *first-price sealed-bid auction*, a bidder who offers the highest price gets a good in the highest price. However, a bidder does not have the *dominant strategy* (optimal strategy) in this auction type, so a winning bid may be much higher or much lower. There are many studies on an electronic first-price sealed-bid auction[2,4,7,8,9,11,12,13,14,15,16,17]. On the other hand, in a *second-price sealed-bid auction*, a bidder who offers the highest price gets a good in the second highest price. This style of auction has the *incentive compatibility*. The dominant strategy for each bidder is to place a bid honestly her/his own true value[18]. So it works the competition principle as well as English auction and a winning bid reflects a market price better than a first-price sealed-bid auction. In our scheme, we electronically realize a second-price sealed-bid auction.

An electronic second-price sealed-bid auction should satisfy the following properties:

* This work has been supported by the Telecommunications Advancement Organization of Japan under the grant for international joint research related to information-communications.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 57–71, 2003.

(a) **Secrecy of the highest bid:** The scheme should not disclose the exact value of the highest bid. Furthermore, nobody can know the information about the highest bid except that it is placed higher than the second highest bid value. This property is desired for secrecy of winner's bid.
(b) **Anonymity of the second highest bid:** Nobody can identify a bidder who places the second highest bid ($\mathcal{B}_{sec}$). This property is desired because the second highest bid is opened.
(c) **Public verifiability:** Anyone can verify the correctness of an auction.
(d) **Secrecy of loosing bids:** The scheme should keep loosing bids secret. This property is desired in order to keep loser's privacy for the auction managers.
(e) **Robustness:** Any malicious bid can be detected and removed justly by authorities.
(f) **Non-cancelability:** A winner cannot deny that she/he submitted the highest bid after the winner decision procedure.

It is easy to apply a second-price sealed-bid auction to a first-price sealed-bid auction. But a first-price sealed-bid auction cannot directly be applied to a second-price sealed-bid auction which keeps the highest bid secret with public verifiability. This is why we need new techniques for a second-price sealed-bid auction.

1.2 Related Works

We discuss several studies[12,6,1] as a second-price sealed-bid auction. These schemes set the bidding points discretely. [12] realizes some kinds of sealed-bid auctions using two auction managers AM1 and AM2, which applies the oblivious transfer. But this scheme requires the cut-and-choose technique in order to satisfy public verifiability. Kikuchi [6] also proposed the (M+1)st-price sealed-bid auction using the verifiable secret sharing technique, where the bidding point is represented by the degree of a polynomial shared by the number of AMs m. In his scheme, there exist some drawbacks: (1) this scheme has a undesirable condition that m is larger than the number of bidding points, so it is difficult to set many bidding points; (2) anyone can anonymously disturb an auction by submitting an invalid bid. These problems are solved in our scheme. Abe and Suzuki [1] proposed the (M+1)st-price sealed-bid auction using homomorphic encryption and mix and match technique[5]. Their scheme realizes public verifiability of a winner and the winning bid. However, each bidder must compute K+1 zero-knowledge proofs in bidding, where K is the number of bidding points.

1.3 Our Result

Our second-price sealed-bid auction scheme uses two kinds of auction managers (AM1 and AM2). AM1 treats the bidder registration. AM2 manages the bidding phase in an auction. Only the cooperation of both AM1 and AM2 can decide a winning bid, together with a winner. In the bidding phase, each bid can be verified by AM1 and AM2. In the opening phase, anyone can verify the auction

process and the results (a winning bid and a winner) by the techniques of the discriminant function of the p_0-th root, the verifiable w-th power mix, the verifiable ElGamal decryption, and the verifiable decryption mix. Our scheme satisfies the above properties. Nobody can know the information about the highest bid except that it is placed higher than the second highest bid value, but anybody can publicly verify the auction results. There is no single entity who knows the highest bid value, a bidder $\mathcal{B}_{sec}$, and loosing bid values by himself. Furthermore, each bidder does not have to compute the zero-knowledge proofs unlike [1]. So the computational cost of bidder is lower.

The remaining of this paper is organized as follows. Section 2 discusses the effect of a second-price sealed-bid auction from the viewpoints of economics. Section 3 reviews the previous scheme[1] and describes its drawbacks. Section 4 describes our protocol in detail. Section 5 investigates the features of our scheme.

2 Economic Viewpoints

2.1 Advantages of a Second-Price Sealed-Bid Auction

A second-price sealed-bid auction has been proposed by W. Vickrey in 1961[18], who won the Nobel Economics Prize in 1996. A second-price sealed-bid auction is that each bidder secretly submits a bid to Auctioneer only once, and a bidder who offers the highest price gets a good in the second highest price. Here we explain why a second-price sealed-bid auction is so outstanding by the following example. Three bidders $\{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$ participate the car, BMW, auction and their *true values* for it, which means the maximum value that each bidder can spend, are as follows:

- $\mathcal{B}_1$'s true value : \$66,000;
- $\mathcal{B}_2$'s true value : \$64,400;
- $\mathcal{B}_3$'s true value : \$60,900.

If a bidder can buy BMW cheaper than her/his true value, she/he will make a profit. If she/he buys BMW higher than her/his true value, her/his purchase will end in failure. So the true value means the boundary between losses and gains for each bidder.

Suppose that they participate in a first-price sealed-bid auction under the above situation. Then each bidder will never place her/his true value because she/he wants to buy BMW as cheap as possible. In this case, it is often happened for each bidder to tap other bids in order to estimate exactly her/his bid since they can buy it as cheap as possible. If a winning bid is much higher than the second highest price, a winner may want to cancel it. Even if a winner bought a good, she/he will not agree with it.

However, suppose that they participate in a second-price sealed-bid auction. Then each bidder will place her/his true value because she/he cannot reduce her/his cost for BMW by her/his bid. Generally, it is said that a bidder has the dominant strategy in a second-price sealed-bid auction. So it is useless for each

bidder to estimate other bids. A winner's bid is decided by other bids. A winner's bid value is not a winning bid value but a datum line to decide a winner. So any bidder will place her/his true value in a second-price sealed-bid auction, which has the following property of incentive compatibility.

Incentive compatibility: Incentive compatibility means that the dominant strategy for each bidder is to place a bid honestly her/his own true value[18].

Each bidder can place a bid through mutual agreement. As a result, a bidder will not want to cancel her/his bid. Therefore a second-price sealed-bid auction is superior to a first-price sealed-bid auction from the view points of economics.

Next we compare a second-price sealed-bid auction with an English auction. A winning bid value in a second-price sealed-bid auction becomes the second highest true value ($64,400) as mentioned above. On the other hand, in an English auction, each bidder places a bid many times until their true value. As a result, $\mathcal{B}_1$ gets BMW in $64,400+$\Delta$ ($\Delta \simeq 0$) since $\mathcal{B}_2$ does not place a bid in more than $64,400. Therefore a winning bid in a second-price sealed-bid auction is almost the same value as one in an English auction. This means that a second-price sealed-bid auction works the competition principle as well as an English auction.

2.2 Disadvantages

We wonder if a second-price sealed-bid auction is superior to English auction. Actually, however, an English auction is much more popular than a second-price sealed-bid auction. We think two reasons why a second-price sealed-bid auction is unpopular as follows:

1. A winning bid value is not winner's.
2. It is hard for each bidder to decide her/his true value in advance.

If the AM knows the highest bid value in the middle of auction, the AM may place a little lower bid than the highest bid as a valid bidder. In this case, a winning bid almost becomes winner's true value. Even a winner does not perceive such AM's handling. As long as the AM knows the highest bid value in the middle of auction, the bidder will not want to participate in the second-price sealed-bid auction. Such AM's handling cannot be happen in English auction. This is why secrecy of the highest bid is necessary for an authority in the second-price sealed-bid auction.

In the case 2, a bidder must decide her/his true value for the dominant strategy in advance. However, the bidder $\mathcal{B}_{sec}$ may change her/his true value in the middle of the auction. The true value depends on bidder's mood whether the bidder wants to buy the good. After an auction, $\mathcal{B}_{sec}$'s true value may be higher than the winner's bid value. Then $\mathcal{B}_{sec}$ may regret her/his bid. In an English auction, a bidder can raise her/his true value in the middle of auction.

3 Previous Scheme

Here we summarize a previous scheme[1] which uses homomorphic encryption and mix and match technique.

3.1 Protocol

There are bidders $\mathcal{B}_1, \ldots, \mathcal{B}_I$, auction manager AM, and the trusted authority TA. The TA generates a secret key and a public key of ElGamal cryptosystem that each bidder uses in the bidding phase. The AM sets the bidding points $\{1, \ldots, K\}$. When a bidder places a bid, she/he generates a bid vector which conceals the bid value by ElGamal encryption E. A bidder must send either E(1) or E(r) as the element of bid vector. The TA can know any bidder's bid value by decrypting the element. In order to conceal the bid values for the TA, this scheme may share the secret key among plural authorities by using a secret sharing technique.

In the opening phase, this scheme uses the following homomorphic property for each bidding point:

$$\overbrace{E(1)\cdots E(1)}^{I-b}\overbrace{E(r)\cdots E(r)}^{b} = E(r^b),$$

where E is an ElGamal encryption and r is public number. Suppose that I is the number of bidders and b is the bidding number in the bidding point k. The mix and much technique can publicly show whether $D^*(E(r^\lambda)) \in \{1, r, r^2, \ldots, r^I\}$ or not, where D^* is the verifiable ElGamal decryption. If $D^*(E(r^\lambda))$ is r^b, b bidders place a bid in the bidding point k. The AM finds the highest bidding point so that $D^*(E(r^\lambda))$ might be r^{M+1}, where M is the number of winners. It becomes the second highest bid (a winning bid value).

3.2 Drawbacks

Since a bidder must send either E(1) or E(r) as the element of bid vector, each bidder must compute K+1 zero-knowledge proofs that each element in bid vector is whether E(1) or E(r). So the computational cost for a bidder gets rather large.

4 Our Scheme

4.1 Goals

Our main goals are to realize the following three requirements in an electronic second-price sealed-bid auction, where $\mathcal{B}_{sec}$ is a bidder who places the second highest bid:

1. The highest bid value are not disclosed for any entity;
2. Anonymity of $\mathcal{B}_{sec}$ is satisfied for any entity;
3. Anyone can publicly verify the auction process and results.

The first goal is desired even after winner's decision in order to satisfy a privacy of winner. Our scheme does not disclose the highest bid value as well as the partial range that the highest bid is placed for any entity including both auction managers (AM1 and AM2). The second goal is important because $\mathcal{B}_{sec}$'s bid is revealed as a winning bid. Our scheme realizes anonymity of $\mathcal{B}_{sec}$ without an anonymous channel. The correspondence of each bid to each bidder is also kept secret unless both AM1 and AM2 collude. The third goal ((c) Public verifiability) is important because our scheme secretly computes the auction results.

Furthermore, in our scheme, each bidder does not have to compute the zero-knowledge proofs unlike [1]. To reduce the computational cost of bidder is one of our goals.

4.2 Authorities

Our scheme uses two kinds of auction managers (AM1 and AM2) in order to eliminate a strong single authority. The role of each auction managers is as follows:

- **AM1:**
  - treats the bidder registration;
  - publicly computes the winning bid, decides a winner, and show the validity of the results;
  - manages AM1's bulletin board system (BBS) which publishes a list of public keys and shows the validity of the results.
- **AM2:**
  - manages the bidding phase;
  - verifies a bid information;
  - publicly multiplies each element in all bid vectors;
  - manages AM2's BBS which publishes the computing process of bids.

4.3 Notations

Notations are defined as follows:

I	: the number of bidders;
i	: the index of bidders;
$\mathcal{B}_i$	: a bidder i $(i = 1, \ldots, I)$;
$\mathcal{B}_{sec}$	: a bidder who places the second highest bid;
$\mathbf{V}_i$	: a bid vector of bidder i;
p_0, p_1	: small primes but greater in bit size than number of bidders, I (e.g.100bit);
p, q, p', q'	: large primes $(p = 2p_0p' + 1, q = 2p_1q' + 1)$ which are secret except for the AM1;
n	: $n = pq$;
g	: $g \in_R \mathbf{Z}_n$ whose order is $\mathrm{ord}(g) = 2p_0p'p_1q'$ and has neither p_0-th nor p_1-th root;
k	: the index of bidding points $(k = 1, \ldots, K)$;

$t_{i,k}^{(0)}$, $t_{i,k}^{(1)}$: $\mathcal{B}_i$'s secret random numbers generated by the AM1;
x_i : $\mathcal{B}_i$'s private key;
y_i : $\mathcal{B}_i$'s public key ($y_i = g^{x_i} \mod n$);
s, w : AM2's private keys (w is relatively prime to p_0 : $\gcd(w, p_0) = 1$);
Y : AM2's public key ($Y = g^s \mod n$) that has neither p_0-th nor p_1-th root;
$sig_{key}()$: a signature by key;
$E_y()$: ElGamal encryption with public key g and $y = g^x$ such as $E_y(m) = (G = g^r, M = my^r)$;
$D^*()$: the verifiable ElGamal decryption
$\mathcal{M}()$: the discriminant function of the p_0-th root, where $\mathcal{M}(y)$ is 1 or 0 whether y has the p_0-th root in $\mathbf{Z}_n$ or not, which can be computed only by the AM1.

4.4 Building Blocks

The ElGamal public-key cryptosystem over $\mathbf{Z}_n$ is as secure as the Diffie-Hellman scheme described in [10]. In this scheme, we summarize some proofs of knowledge[3] and their applications over $\mathbf{Z}_n$.

Proof of knowledge. We present three kinds of signatures based on a proof of knowledge.

- $SPK[(\alpha) : y_1 = g_1^\alpha \wedge y_2 = g_2^\alpha](m)$: the proof of the equality of two discrete logarithms.
- $SPK[(\alpha, \beta) : y_1 = g_1^\alpha \vee y_2 = g_2^\beta](m)$: the proof of the knowledge of one out of two discrete logarithms.
- $SPK[(\alpha, \beta) : (y_1 = g_1^\alpha \wedge y_3 = g_3^\alpha) \vee (y_2 = g_2^\beta \wedge y_3 = g_3^\beta)](m)$: the proof of the knowledge of one out of two discrete logarithms, which is equal to another discrete logarithm of y_3 to the base g_3. This SPK is given by combining above two SPKs.

The Verifiable p_0-th Root

Lemma 1. *For $n = pq$ ($p = 2p'p_0 + 1, q = 2q' + 1, p', q', p_0$: different primes > 2), $z \in \mathbf{Z}_n$ has the p_0-th root if and only if $z^{2p'q'} = 1 \pmod n$.*

Proof. If z has the p_0-th root, there exists x such that $z = x^{p_0}$. Therefore, $z^{2p'q'} = x^{2p'p_0q'} = 1 \pmod n$. Conversely, if $z^{2p'q'} = 1 \pmod n$, then there exists x such that $z^{2p'q'} = x^{2p'p_0q'} \pmod n$ (order of x is $2p'p_0q'$). Therefore, $z = x^{p_0} \pmod n$, see, z has the p_0-th root. □

$M(z)$ can be computed by only the knowledge of p' and q'. Therefore an authority who knows order of g can publicly prove that z has the p_0-th root by showing

$$SPK[(\alpha) : z^\alpha = 1 \wedge (g^{p_0})^\alpha = 1 \wedge g^\alpha = r](m),$$

for a random number $r \neq 1$. Also, such an authority can publicly prove that z does not have the p_0-th root by showing

$$SPK[(\alpha) : z^{\alpha} = r \wedge (g^{p_0})^{\alpha} = 1](m),$$

for random numbers $r \neq 1$. The above two SPKs mean that α is $2p'q'$. Checking whether z has the p_0-th root or not satisfies public verifiability.

Verifiable w-th power mix. A pair of $(c, C = c^w)$ is published, where w is secret. Let (a, b) and (A, B) be input and output of the verifiable w-th power mix, respectively, where $A = a^w$ and $B = b^w$ $(A \neq B)$. We hide the correspondence of an input to the output, but can show the validity of secret mix by proving the equality of three discrete logarithms of A, B and C. The proof is given by showing

$$SPK[(\alpha) : (A = a^{\alpha} \wedge B = b^{\alpha} \wedge C = c^{\alpha}) \vee (A = b^{\alpha} \wedge B = a^{\alpha} \wedge C = c^{\alpha})](m).$$

Verifiable ElGamal decryption. We can prove that $m = M/G^s$ is the decryption of $E_Y(m) = (G, M)$ without revealing s by showing

$$SPK[(\alpha) : M/m = G^{\alpha} \wedge Y = g^{\alpha}](m).$$

Verifiable decryption mix. Let $(E_Y(a), E_Y(b))$ and (a, b) be input and output of the verifiable decryption mix, respectively, where $E_Y(a) = (G_a, M_a)$ and $E_Y(b) = (G_b, M_b)$. We hide the correspondence of an input to the output, but can show the validity of secret mix. The proof is given by showing

$$\begin{aligned} SPK[(\alpha) : &(M_a/a = {G_a}^{\alpha} \wedge M_b/b = {G_b}^{\alpha} \wedge Y = g^{\alpha}) \\ &\vee (M_a/b = {G_a}^{\alpha} \wedge M_b/a = {G_b}^{\alpha} \wedge Y = g^{\alpha})](m). \end{aligned}$$

4.5 Procedure

[Initialization:] The AM1 selects g, p_0, p_1, p', q', p and q, computes a product $n = pq$, and then publishes (g, p_0, p_1, n) but keeps (p', q', p, q) secret. The AM1 also sets the number K of bidding points for a good. The AM2 computes $Y = g^s \pmod{n}$ and publishes Y. Note that s is AM2's secret and that both $\gcd(s, p_0) = 1$ and $\gcd(s, p_1) = 1$ hold. The AM1 checks that Y has neither the p_0-th nor p_1-th root and that order of Y is $2p_0p'p_1q'$.

[Bidder registration:] When Alice $(\mathcal{B}_i)$ participates an auction, she sends her public key y_i with the signature $sig_{x_i}(y_i)$ to the AM1 as a bidder registration. After the AM1 receives her values, he publishes her public key y_i.

[Auction preparation:] The AM1 chooses her values $t_{i,1}^{(0)}, \ldots, t_{i,K}^{(0)}, t_{i,1}^{(1)}, \ldots t_{i,K}^{(1)} \in \mathbf{Z}_n$, all of which have the p_0-th root, and then secretly sends $\{t_{i,k}^{(0)} \cdot g^{p_0}\}$ and $\{t_{i,k}^{(1)} \cdot g^{p_1}\}$ to $\mathcal{B}_i$. The AM1 shuffles two values in every bidding point:

$$\left(\mathcal{H}(t_{i,1}^{(0)} \cdot g^{p_0}), \mathcal{H}(t_{i,1}^{(1)} \cdot g^{p_1})\right), \ \ldots, \ \left(\mathcal{H}(t_{i,K}^{(0)} \cdot g^{p_0}), \mathcal{H}(t_{i,K}^{(1)} \cdot g^{p_1})\right),$$

for $i = 1, \ldots, I$, and places them into AM1's public database. By checking AM1's public database, $\mathcal{B}_i$ can confirm whether her values $t_{i,1}^{(0)} \cdot g^{p_0}$, $\ldots$, $t_{i,K}^{(0)} \cdot g^{p_0}$, $t_{i,1}^{(1)} \cdot g^{p_1}$, $\ldots$, $t_{i,K}^{(1)} \cdot g^{p_1}$ are exactly registered. We assume that: nobody except the AM1 knows the correspondence of a bidder to her/his two values; anybody can refer to the data in his public database; but that only the AM1 can alter them.

[Bidding:] When Alice places a bid at a bidding point $k_i \in \{1, \ldots, K\}$, she generates her bid vector $\mathbf{V}_i$ as follows:

$$\mathbf{V}_i = [E_Y(v_{i,K}), \ldots, E_Y(v_{i,1})],$$

where

$$v_{i,k} = \begin{cases} t_{i,k}^{(1)} \cdot g^{p_1} \pmod{n} & (k = k_i), \\ t_{i,k}^{(0)} \cdot g^{p_0} \pmod{n} & (k \neq k_i). \end{cases}$$

She sends $\mathbf{V}_i$ to the AM2. Note that she also sends her reverse bid vector $\mathbf{V}_i' = [E_Y(v_{i,K}'), \ldots, E_Y(v_{i,1}')]$, see, if $v_{i,k} = t_{i,k}^{(0)} \cdot g^{p_0}$, then $v_{i,k}' = t_{i,k}^{(1)} \cdot g^{p_1}$.

[Checking a bid vector:] The validity of $\mathbf{V}_i$ is checked as follows: (1) The AM2 decrypts $\{E(v_{i,k}), E(v_{i,k}')\}$ by using the verifiable decryption mix; (2) The AM2 computes both $\mathcal{H}(v_{i,k})$ and $\mathcal{H}(v_{i,k}')$ and checks whether or not both values exist in AM1's public database; (3) The AM2 computes

$$\Gamma 1_i = \frac{1}{g^{p_1}} D^* \left(\prod_{k=1}^{K} E_Y(v_{i,k}) \right) \quad \text{and} \quad \Gamma 2_i = \frac{1}{g^{Kp_1}} \prod_{k=1}^{K} v_{i,k} v_{i,k}' \qquad (i = 1, \ldots, I)$$

by using the verifiable decryption D^*; (4) The AM1 publicly shows that both $\Gamma 1_i$ and $\Gamma 2_i$ have the p_0-th root. Thanks to this confirmation, any malicious bid vector can be detected by the cooperation of AM1 and AM2. Note that the AM2 does not know whether $v_{i,k}$ and $v_{i,k}'$ have the p_0-th root or not.

[Opening a winning bid:] First, a winning bid is decided, then a winner is decided by the cooperation of both AM1 and AM2.

Step 1 The AM2 publicly computes the following values for $\mathcal{B}_i$:

$$E_Y(z_{i,K}), E_Y(z_{i,K-1}), \ldots, E_Y(z_{i,1}) = E_Y(v_{i,K}), E_Y(v_{i,K} v_{i,K-1}), \ldots, E_Y(\prod_{k=1}^{K} v_{i,k}).$$

for $i = 1, \ldots, I$, and then puts them in AM2's BBS.

Step 2 The AM2 publicly computes the following two kinds of values by multiplying $E_Y(z_{i,k})$ of all bidders for a bidding point k,

$$E_Y(Z_k) = \prod_{i=1}^{I} E_Y(z_{i,k}) = \left(g^R, \left(\prod_{i=1}^{I} z_{i,k}\right) \cdot Y^R\right) = (G_k, M_k),$$

$$E_Y(Z'_k) = \left(g^R, \frac{1}{g^{p_1}}\left(\prod_{i=1}^{I} z_{i,k}\right) \cdot Y^R\right) = (G_k, M'_k) \qquad k \in \{1, \ldots, K\},$$

where R is the sum of all bidder's random numbers in ElGamal encryption.

Step 3 The AM2 mixes $(E_Y(Z_k), E_Y(Z'_k))$ into $((E_Y(Z_k))^w, (E_Y(Z'_k))^w)$ using w relatively prime to p_0 and the technique of the verifiable w-th power mix, and then publishes the following values:

$$(E_Y(Z_k))^w = E_Y({Z_k}^w) = ({G_k}^w, {M_k}^w),$$
$$(E_Y(Z'_k))^w = E_Y({Z'_k}^w) = ({G_k}^w, {M'_k}^w).$$

The AM1 can publicly show that w is relatively prime to p_0 by using the verifiable w-th power mix in 4.4.

Step 4 The AM2 decrypts $E_Y({Z_k}^w)$ and $E_Y({Z'_k}^w)$ into $\mathcal{X}_k = {Z_k}^w$ and $\mathcal{Y}_k = {Z'_k}^w$ using the technique of the verifiable decryption, and publishes $(\mathcal{X}_k, \mathcal{Y}_k)$.

Step 5 The AM1 computes $\mathcal{M}(\mathcal{X}_k)$ and $\mathcal{M}(\mathcal{Y}_k)$, and publishes a tuple of $(\mathcal{X}_k, \mathcal{Y}_k, \mathcal{M}(\mathcal{X}_k), \mathcal{M}(\mathcal{Y}_k))$. A winning bid value is given by the highest bidding point with both $\mathcal{M}(\mathcal{X}_k) = 0$ and $\mathcal{M}(\mathcal{Y}_k) = 0$.

Since the values $\{t_{i,k}^{(0)}, t_{i,k}^{(1)}\}$ have the p_0-th root, g has neither p_0-th nor p_1-th root, and $\gcd(w, p_0) = 1$ holds, the following three cases are occurred for the values of $\mathcal{M}(\mathcal{X}_k)$ and $\mathcal{M}(\mathcal{Y}_k)$ in Figure 1:

1. If no bidder places a bid equal to or higher than the bidding point k, then $(\mathcal{M}(\mathcal{X}_k),\ \mathcal{M}(\mathcal{Y}_k)) = (1, 0)$.
2. If only one bidder places a bid equal to or higher than the bidding point k, then $(\mathcal{M}(\mathcal{X}_k),\ \mathcal{M}(\mathcal{Y}_k)) = (0, 1)$.
3. If more than two bidders place a bid equal to or higher than the bidding point k, then $(\mathcal{M}(\mathcal{X}_k),\ \mathcal{M}(\mathcal{Y}_k)) = (0, 0)$.

Note that we cannot distinguish between case 1 and case 2 because the AM2 uses the technique of the verifiable w-th power mix for $\mathcal{X}_k$ and $\mathcal{Y}_k$.

Public verifiability of a winning bid: The AM1 may rig a winning bid because only the AM1 computes $\mathcal{M}(\mathcal{X}_k)$ and $\mathcal{M}(\mathcal{Y}_k)$. In order to avoid rigging, the AM1 shows the following SPK:

$$SPK[(\alpha) : \mathcal{X}_k^\alpha = r_1 \wedge \mathcal{Y}_k^\alpha = r_2 \wedge \mathcal{X}_{k+1}^\alpha = r_3 \wedge \mathcal{Y}_{k+1}^\alpha = 1](m)$$

for given random numbers r_1, r_2 and r_3 $(r_1, r_2, r_3 \neq 1)$. This SPK means that only $\mathcal{Y}_{k+1}$ has the p_0-th root.

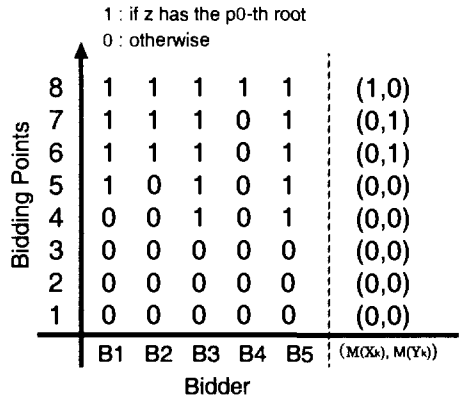


Fig. 1. Opening Example

Furthermore, the cost of opening bids is $O(\log K)$ by adopting the technique introduced in [4,6]: (1) For a set of bidding points $\{1, \ldots, K\}$, set $k_1 = 1, k_2 = K$ and $k' = \lfloor \frac{k_1+k_2}{2} \rfloor$; (2) If $k' = k_1$ or $k' = k_2$, then output k_2 as the second highest bid value; (3) If $\mathcal{M}(\mathcal{X}_{k'}) = 0$ and $\mathcal{M}(\mathcal{Y}_{k'}) = 0$, then set $k_1 = k'$ and $k' = \lfloor \frac{k_2+k'}{2} \rfloor$, and go to (2). Otherwise set $k_2 = k'$ and $k' = \lfloor \frac{k_1+k'}{2} \rfloor$, and go to (2).

[Winner decision:] After a winning bid value k (the second highest bid) is decided, the AM2 decrypts all the values $v_{i,k+1}$ $(i = 1, \ldots, I)$ using the technique of the verifiable decryption. Anyone can confirm whether or not these values $v_{i,k+1}$ $(i = 1, \ldots, I)$ exist in AM1's BBS.

Public verifiability of a winner: In order to decide a winner $\mathcal{B}_j$, the AM1 shows the following SPK:

$$SPK[(\alpha) : (g^{p_0})^{\alpha} = 1 \wedge (v_{j,k+1})^{\alpha} = r_1](m)$$

for given random number r_1 $(r_1 \neq 1)$. This SPK means that $v_{j,k+1}$ does not have the p_0-th root. A winner $\mathcal{B}_j$'s bid is never revealed. If no bidder places a bidding point $k+1$, more than two winners place a bid at the bidding point k. This means that a winning bid is also k. The AM1 shows the following SPK:

$$SPK[(\alpha) : g^{\alpha} = r_2 \wedge (v_{1,k+1})^{\alpha} = 1 \wedge \cdots \wedge (v_{I,k+1})^{\alpha} = 1](m)$$

for given random number r_2 $(r_2 \neq 1)$. This SPK means that all values $v_{i,k+1}$ $(i = 1, \ldots, I)$ have the p_0-th root. Note that g does not have the p_0-th root.

5 Consideration

5.1 Features

We discuss the following properties in our protocol.

(a) **Secrecy of the highest bid:** Our scheme keeps the highest bid secret unless both the AMs collude. Nobody can know the information about the highest bid except that it is placed higher than the second highest bid value. Each element $v_{i,k}$ $(z_{i,k})$ has information about whether it has the p_0-th root or not. So only AM1 who knows the products of n realizes the bid values from the values $v_{i,k}$ $(z_{i,k})$. However, such a bid value is encrypted by ElGamal encryption of AM2, and the values $v_{i,k}$ $(z_{i,k})$ themselves are never revealed in the auction procedure. Therefore, AM1 cannot know bid values as long as the ElGamal encryption is secure. Also, AM2 cannot realize bid values because she/he does not know the products of n, even if AM2 knows the values $v_{i,k}$ $(z_{i,k})$. By applying the verifiable w-th power mix to step 3 of the opening phase, the highest bid value can be hidden. Since the AM1 can publicly show that w is relatively prime to p_0, the highest bid value remains correct.

(b) **Anonymity of the second highest bid:** Unless both of the AMs collude, nobody can identify the bidder $\mathcal{B}_{sec}$ even if an anonymous channel is not used. Since all bid vectors are multiplied together before the opening phase, the bidder $\mathcal{B}_{sec}$ is never disclosed. If all bid values are disclosed in the bidding phase, the bidder $\mathcal{B}_{sec}$ is easily decided. As described in (a), each bid value is protected by both hardness of the discriminant of the p_0-th root and the ElGamal encryption. So the identity of $\mathcal{B}_{sec}$ can be protected without using an anonymous channel.

(c) **Public verifiability:** Anyone can publicly verify the correctness of an auction. An auction uses some tools based on the proof of knowledge in order to satisfy public verifiability. As long as the proofs of knowledges are secure, an auction process can be collect. As a result, both a winning bid and a winner become valid.

(d) **Secrecy of loosing bids:** Our scheme keeps loosing bids secret unless both of AMs collude. This feature can be discussed similar to (a).

(e) **Robustness:** Any malicious bid vector can be detected by AM1 and AM2. Unless a bidder uses the valid $v_{i,k}$ and $v'_{i,k}$, anybody notices that $H(v_{i,k})$ or $H(v'_{i,k})$ does not exist in AM1's database. Also, unless a bidder generates the valid V_i, the AM1 notices that $\Gamma 1_i$ and $\Gamma 2_i$ do not have the p_0-th root

Table 1. The communicational costs

	A bidder ($\mathcal{B}$)	AM		
	Bidding	Preparation	Opening × Round	#AM
[AS02]	$O(K)$	–	$O(1) \times \lceil \log K \rceil$	2
Ours	$O(K)$	$O(IK)$	$O(1) \times \lceil \log K \rceil$	2

Table 2. The computational costs (bidder)

	#Enc	#Proof
[AS02]	K	$K+1$
Ours	$2K$	–

after the AM2 computes them. So no bidder can disturb the auction system by the malicious bid.

(f) **Non-cancelability:** A winner cannot deny that she/he has submitted the highest bid after the winner decision procedure as long as both (c) and (e) are satisfied. Since the AM1 publicly shows the SPK(s) for the winner decision, a winner is certainly identified.

(g) **Two independent AM's powers:** Our scheme is based on both RSA and ElGamal cryptosystems. Only the AM1 knows the prime factors of n, while only the AM2 knows the secret key of ElGamal encryption. Thanks to separation of two kinds of the cryptosystems, neither AM1 nor AM2 knows the highest bid value, a bidder $\mathcal{B}_{sec}$, and loosing bid values.

5.2 Efficiency

We compare our scheme with the previous scheme[1] from the viewpoints of the communicational and computational costs in Table 1, 2 and 3. Here let the number of bidding points and bidders be K and I, respectively.

Table 1 shows the communicational amount of bidding and between the AMs. In both [1] and our scheme, only $\lceil \log K \rceil$ rounds of communication are required in the opening phase because of binary search. In the auction preparation of our scheme, the AM1 must send K ElGamal encryption data to each bidder.

Table 2 and 3 show the computational complexity. In [1], each bidder requires the $K+1$ proofs to avoid the malicious bidding. In our scheme, each bidder does not need to make such proofs, but the AM2 generates $K+1$ proofs for I bidders. In [1], the AM needs the bid checking of the cost $O(IK)$ in order to verify the proofs. In our scheme, the AM2 needs the bid checking of the cost only $O(I)$ because it uses the sum of all bid vectors. The AM1 needs IK ElGamal encryptions for an auction preparation. As for the number of decryption, our scheme requires $2IK$ times in generating proofs, I times in the bid checking, $2\lceil \log K \rceil$ times in the opening phase, and I times in the winner decision phase.

If [1] applies the secret sharing technique for the sake of the TA distribution, both communicational and computational costs becomes larger.

Table 3. The computational costs (AM)

	#Enc	#Proof	#Multiplication	Bid check	#Dec
[AS02]	–	–	$IK + I\lceil \log K \rceil$	$O(IK)$	$2\lceil \log K \rceil + I$
Ours	IK	$I(K+1)$	$2(IK + I\lceil \log K \rceil)$	$O(I)$	$2\lceil \log K \rceil + 2I(K+1)$

6 Conclusion

We have proposed an electronic second-price sealed-bid auction which mainly satisfies (a) Secrecy of the highest bid, (b) Anonymity of the second-price bid, (c) Public verifiability, and (g) Two independent AM's powers. In our scheme, there is no single entity who knows the highest bid value, a bidder $\mathcal{B}_{sec}$, and loosing bid values. Also, each bidder does not have to compute the zero-knowledge proofs, but the AM computes such proofs. So the computational cost of bidder is lower.

References

1. M. Abe and K. Suzuki. "M+1-st Price Auction Using Homomorphic Encryption". In *Proceedings of the 5-th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002)*, LNCS, Springer-Verlag, page to appear, 2002.
2. C. Cachin. "Efficient Private Bidding and Auctions with an Oblivious Third Party". In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 120–127, 1999.
3. J. Camenisch and M. Michels. "A Group Signature Scheme with Improved Efficiency". In *Advances in Cryptology - ASIACRYPT '98*, LNCS 1514, Springer-Verlag, pages 160–174, 1998.
4. K. Chida, K. Kobayashi, and H. Morita. "Efficient Sealed-bid Auctions for Massive Numbers of Bidders with Lump Comparison". In *Proceedings of the 4th Information Security Conference (ISC 2001)*, LNCS 2200, Springer-Verlag, pages 408–419, 2001.
5. M. Jakobsson and A. Juels. "Mix and Match: Secure Function Evaluation via Ciphertexts". In *Advances in Cryptology - ASIACRYPT 2000*, LNCS 1976, Springer-Verlag, pages 162–177, 2000.
6. H. Kikuchi. "(M+1)st-Price Auction Protocol". In *Proceedings of the 5th International Financial Cryptography (FC 2001)*, LNCS, Springer-Verlag, page to appear, 2001.
7. H. Kikuchi, M. Harkavy, and D. Tyger. "Multi-round anonymous auction protocols". In *Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69, 1998.
8. K. Kobayashi, H. Morita, K. Suzuki, and M. Hakuta. "Efficient Sealed-bid Auction by Using One-way Functions". *IEICE Trans. Fundamentals*, Vol.E84-A,No.1:pp.289–294, 2001.
9. M. Kudo. "Secure electronic sealed-bid auction protocol with public key cryptography". *IEICE Trans. Fundamentals*, Vol.E81-A,No.1:pp.20–27, 1998.
10. M. Manbo and H. Shizuya. "A Note on the Complexity of Breaking Okamoto-Tanaka ID-Based Key Exchange Scheme". *IEICE Trans. Fundamentals*, Vol.E82-A,No.1:77–80, 1999.
11. T. Nakanishi, T. Fujiwara, and H. Watanabe. "An Anonymous Bidding Protocol without Any Reliable Center". *Trans. IPS Japan*, Vol.41,No.8:pp.2161–2169, 2000.
12. M. Naor, B. Pinkas, and R. Sumner. "Privacy Preserving Auctions and Mechanism Design". In *Proceedings of ACM Conference on Electronic Commerce*, pages 120–127, 1999.
13. K. Omote and A. Miyaji. "An Anonymous Auction Protocol with a Single Non-trusted Center using Binary Trees". In *Proceedings of Information Security Workshop (ISW 2000)*, LNCS 1975, Springer-Verlag, pages 108–120, 2000.

14. K. Omote and A. Miyaji. "An Anonymous Sealed-bid Auction with a Feature of Entertainment". *Trans. IPS Japan*, Vol.42,No.8:pp.2049–2056, 2001.
15. K. Sako. "An Auction Protocol Which Hides Bids of Losers". In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000)*, LNCS 1751, Springer-Verlag, pages 422–432, 2000.
16. K. Sakurai and S. Miyazaki. "An Anonymous Electronic Bidding Protocol Based on a New Convertible Group Signature Scheme". In *Proceedings of the 5th Australasian Conference on Information and Privacy (ACISP 2000)*, LNCS 1841, Springer-Verlag, pages 385–399, 2000.
17. K. Suzuki, K. Kobayashi, and H. Morita. "Efficient Sealed-bid Auction Using Hash Chain". In *Proceedings of the Third International Conference on Information Security and Cryptology (ICISC 2000)*, LNCS 2015, Springer-Verlag, pages 189–197, 2000.
18. W. Vickrey. "Counter Speculation, Auctions, and Competitive Sealed Tenders". *Journal of Finance*, Vol.16:pp.8–37, 1961.

A Two-Server, Sealed-Bid Auction Protocol

Ari Juels and Michael Szydlo

RSA Laboratories
Bedford, MA 01730, USA
{ajuels,mszydlo}@rsasecurity.com

Abstract. Naor, Pinkas, and Sumner introduced and implemented a sealed-bid, two-server auction system that is perhaps the most efficient and practical to date. Based on a cryptographic primitive known as oblivious transfer, their system aims to ensure privacy and correctness provided that at least one auction server behaves honestly. As observed in [19], however, the NPS system suffers from a security flaw in which one of the two servers can cheat so as to modify bids almost arbitrarily and without detection. We propose a means of repairing this flaw while preserving the attractive practical elements of the NPS protocol, including minimal round complexity for servers and minimal computation by players providing private inputs. Our proposal requires a slightly greater amount of computation and communication on the part of the two auction servers, but actually involves much *less* computation on the part of bidders. This latter feature makes our proposal particularly attractive for use with low-power devices. While the original proposal of NPS involved several dozen exponentiations for a typical auction, ours by contrast involves only several dozen modular multiplications.
The key idea in our proposal is a form of oblivious transfer that we refer to as *verifiable proxy oblivious transfer* (VPOT).

Keywords: auction, sealed-bid auction, oblivious transfer, secure multiparty computation, secure function evaluation

1 Introduction

Cryptography offers a broad range of tools for distributing trust among computing entities in flexible and often unexpected ways. In an electronic auction setting, for example, a foundational cryptographic procedure known as *secure function evaluation* enables the submission and processing of sealed bids without the presence of a single, trusted auctioneer. As secure function evaluation is rather impractical in its general form, a large body of research, e.g., [1,5,11,17, 19,24], has focused on tailoring cryptographic protocols specifically to achieve efficient sealed-bid auction systems.

A recent architecture proposed and implemented by Naor, Pinkas, and Sumner [20] represents substantial progress toward the goal of practical sealed-bid auctioning with distributed trust. In their system, bidders submit encrypted bids to a front-end server known as an *auctioneer*. With the involvement of a second,

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 72–86, 2003.

back-end server known as an *auction issuer*, any type of sealed-bid auction may be conducted, e.g., highest-bid auctions, Vickrey auctions, and so forth. The architecture distributes trust between the two servers in the following simple model: If at least one server is honest, the bids of all participants will remain private, and any auction outcome is assured to be correct. There is no robustness, however, in the sense that either server can cause the protocol to terminate. NPS report good scalability, claiming that their system can accommodate thousands of bidders with reasonable overhead. The computational requirement for bidders in their system is approximately one modular exponentiation per bit in a bid representation. See [20] for further details.

As identified in a footnote in work by Jakobsson and Juels [19], however, the NPS system has a serious flaw that permits tampering by one of the servers. Although not explicated in [19], it is easy to see that the auction issuer can modify any bit in any bid without detection. The underlying problem is a variant introduced by NPS on a cryptographic primitive known as *1-out-of-2 oblivious transfer* (1-2 OT), as we now explain.

Basic 1-2 OT is a procedure involving two players, a *Chooser* and a *Sender*. The Sender possesses a pair of values (t_0, t_1). We refer to these values throughout our paper as *tags*. The Chooser elects to receive from the Sender one of these two tags t_b for $b \in \{0,1\}$. The 1-2 OT procedure is oblivious in the sense that the Sender learns negligible information about b. An additional privacy property of 1-2 OT that the Chooser learns negligible information about t_{1-b}, i.e., the value that she did not select. NPS consider a variant on 1-2 OT, called *proxy oblivious transfer*. This variant involves an intervening third party known as a *Proxy*, who receives the value t_b on behalf of the Chooser, but herself learns negligible information about b and t_{1-b}. We provide details on the protocol below. Proxy oblivious transfer accomplishes the goal for which it was designed, namely privacy protection, but does not include any mechanism for *verifiability*. In particular, the proxy oblivious transfer protocol does not ensure that the Sender transmitted t_b as desired. In the NPS auction setting, the Sender (auction issuer) can substitute the tag t_{1-b}. Thus the auction issuer can tamper with bids!

In this paper, we introduce a protocol called *verifiable proxy oblivious transfer* (VPOT) that addresses the vulnerability in the NPS protocol. In principle, introducing verifiability into proxy oblivious transfer is not difficult using basic – and potentially expensive – cryptographic techniques such as zero-knowledge proofs. Our contribution in the design of VPOT is a collection of techniques that render the verification process computationally inexpensive and yet, at the same time, provably secure. When VPOT is introduced into the NPS auction protocol, it increases the computational burden on auction servers somewhat, but actually results in much *less* computation for bidders. This is particularly desirable given the fact that bidders in many settings may wish to employ low-power, handheld devices. Thus, VPOT not only remedies the security flaw in the NPS architecture, but renders the system even more practical.

1.1 Background: Two-Party Computation and the NPS Protocol

Secure function evaluation (also known as secure multi-party computation) began with the work of Yao [26], and Goldreich, Micali, and Wigderson [15], and has spawned an ever growing body of research papers. See [13] for a good overview of early work. The general goal of secure multi-party computation is to enable m players to apply a function F to respective private inputs $X_1, X_2, \ldots, X_m$ such that some subset of players learns $F(X_1, X_2, \ldots, X_m)$, but no player learns additional, non-negligible information. Privacy and robustness against active (indeed, adaptive) adversaries are possible to achieve provided that the adversary controls at most a fraction of the players. Assuming the existence of a broadcast channel, this fraction is $1/2$; otherwise, it is $1/3$. For some recent work aiming to achieve practical multi-party protocols, see, e.g., [8,18].

The two-player case of secure function evaluation is distinguished by its relative simplicity and practicality. The original secure function evaluation protocol of Yao [26] treated this case, and remains an important tool even now. In contrast to more general techniques, in which field operations such as addition and multiplication are the atomic unit of computation, the Yao protocol involves direct computation on boolean gates. While this is a limitation in the general case, many real-world protocols such as auctions involve intensive bitwise manipulation such that boolean circuits are in fact a natural form of representation for the required functions. The Yao protocol is appealing for other reasons as well. Provided that only one player is to learn the output, it is in fact possible to execute the Yao protocol with only one-round of interaction, an observation first set forth implicitly in [20] and explored in detail in [7]. While constant-round secure function evaluation is possible for multiple players, it requires both high overhead and the availability of a broadcast channel [3]. A model in which both players in the Yao protocol learn the output of such computation in a fair manner (given a passive trusted entity) is also possible, and is explored in [6].

Suppose that Alice and Bob wish to engage in the Yao protocol on respective private inputs X_A and X_B such that Bob learns the output $y = F(X_A, X_B)$. Alice constructs a "garbled" circuit representing F. She sends this circuit to Bob, along with a "garbled" representation of X_A. In order to evaluate the "garbled" circuit, Bob additionally needs a "garbled" version of his input X_B. He obtains this from Alice using some form of 1-2 *oblivious transfer* (OT) [21] protocol. This is the component of the Yao protocol that we focus on in detail in this paper. In the case where Alice may cheat, another important component in two-player secure function evaluation protocols are proofs of correct construction of the Yao circuits. A cut-and-choose protocol for this is proposed in [20], while [7] explores use of general non-interactive proof techniques. (If Alice wishes Bob to send y to her such that she can verify its correctness, she need merely embed a verification tag in her "garbled" version of F in the appropriate manner.)

Numerous variants of oblivious transfer have been considered in the literature [13]. Notions of combining bit commitment with oblivious transfer in a theoretical setting to achieve a committed or verifiable oblivious transfer are explored for example in [10] and [9]. These works explore theoretical approaches that treat

oblivious transfer and bit commitment as black boxes, and are thus necessarily expensive. An alternative proposal makes use of a trusted initializer [22]. The key observation made by NPS in their auction system design is that by involving a Proxy in the oblivious transfer procedure, it is possible to expand application of basic Yao style two-server function evaluation so as to accept inputs from an arbitrarily large number of players, i.e., bidders, while the evaluation process is restricted to two auction servers.

Briefly stated, the NPS construction is as follows. The auction issuer (again, the back-end server) constructs a "garbled" representation of a function F that describes the auction protocol. The auctioneer (again, the front-end server) evaluates the circuit for F using "garbled" inputs representing the bids. In order to obtain the "garbled" input for a bit b in a given bid, it is necessary to invoke the proxy oblivious transfer protocol. In this protocol, the bidder plays the role of the Chooser, the auctioneer plays the role of the Proxy, and the auction issuer plays the role of the Sender. The Sender transmits "garbled" inputs (t_0, t_1) for the circuit corresponding to a '0' bit and a '1' bit in the bid. The Chooser selects t_b, which the Proxy receives through the transfer protocol. Having done this for all bits in all bids, the Proxy is able to evaluate F on the input bids and determine the outcome of the auction. The privacy properties of the proxy oblivious transfer protocol ensure that the Proxy does not learn b or t_{1-b} for any bit. The Proxy therefore does not learn any bidding information and can only evaluate F on correct bid amounts. Likewise, the Sender does not learn the bid amounts. Only if the Proxy and Sender collude is this privacy guarantee breached.

NPS include other security enhancements in the protocol. For example, for the auctioneer to ensure that the auction issuer has constructed F correctly, the two engage in a cut-in-choose protocol. Thus, the auctioneer in fact evaluates multiple, independent circuits representing F. We provide more details below.

1.2 Our Contribution: Verifiable Proxy Oblivious Transfer (VPOT)

The failing in the NPS protocol is that the auction issuer can transmit t_{1-b} instead of t_b without detection. To address this problem, we propose a protocol known as verifiable proxy oblivious transfer (VPOT). VPOT enables the Proxy (auctioneer) to ensure that the Sender (auction issuer) sent t_b, as required. VPOT retains all of the privacy characteristics of proxy oblivious transfer.

Here is a simplified overview of VPOT. The Sender provides commitments $\mathcal{C}_0$ and $\mathcal{C}_1$ to tags t_0 and t_1 (respectively representing a '0' bit and '1' bit in a bid). These commitments take the form of a randomly ordered pair $(\mathcal{C}_a, \mathcal{C}_{1-a})$, i.e., a is a randomly selected bit. The Sender also provides a commitment $E[a]$ to ordering a. Observe that the triple $(\mathcal{C}_0, \mathcal{C}_1, E[a])$ binds the Sender to values for t_0 and t_1.

As usual in a 1-2 OT protocol, the Chooser selects a value t_b to be decommitted by the Sender. The Chooser in fact splits this bit b into two shares b_P and b_S such that $b = b_P \oplus b_S$. The Chooser sends the share b_S to the Sender. This is transmitted (via the Proxy) as a ciphertext $E[b_S]$. She sends the share b_P to the Proxy, also in a specially committed form that we do not describe here.

It is the splitting of b into two shares that ensures privacy with respect to the two auction servers (provided that there is no collusion).

Finally, the Chooser transmits to the Proxy a secret value x that enables the Proxy to receive the selected tag t_b. The Sender decommits t_b for the Proxy, who then checks the correctness of the decommitment.

Here is a list of the more interesting cryptographic building blocks used in the construction of VPOT. While none is individually novel *per se*, our new constructions combine them in a novel way, providing a new fundamental building block useful for securely extending traditional two-party techniques to settings with multiple contributors.

- *Double commitment:* The Sender's commitment $C_k(t)$ on tag t in fact consists of a pair of values (Y_1, Y_2). The first value, Y_1, is the commitment on a key or witness k. In particular here, $Y_1 = H(k^3)$, where the cubing operation takes place over an RSA modulus provided by the Sender (as discussed in more detail below). H here is a hash function (modelled as a random oracle for security proofs on the system). Observe that as the hash of a cube, Y_1 is really a commitment within a commitment. It is for this reason that we refer to $C_k(t)$ as a *double* commitment. The second value of the commitment pair, Y_2, represents an encryption of t under k. In particular, $Y_2 = H(k) \oplus t$, where $\oplus$ denotes the bitwise XOR operator. Knowledge of the witness k is sufficient both to open the commitment and obtain t and also to verify that the commitment has been correctly opened. This double commitment scheme may be seen to be both computationally binding and computationally hiding under the RSA assumption, with the random oracle model invoked for H.
- *RSA-based oblivious transfer:* Most oblivious transfer protocols designed for practical use in two-party secure function evaluation, e.g., in [20,2], employ El Gamal-based encryption of tags [14]. The result is that the Chooser must perform at least one exponentiation per 1-2 OT invocation. In contrast, we introduce an RSA-based 1-2 OT scheme as the foundation for VPOT. The result is that the Chooser need only perform one RSA cubing, i.e., two modular multiplications, per 1-2 OT invocation. When employed in VPOT, this idea reduces the work of the Chooser by over an order of magnitude with respect to the proxy oblivious transfer protocol of NPS.
- *Goldwasser-Micali encryption:* The encryption function E in our brief description above is the Goldwasser-Micali cryptosystem [16]. Encryption in this system takes place with respect to an RSA modulus n. A '0' bit is encrypted as a random quadratic non-residue over Z_n, while a '1' bit is encrypted as a random quadratic residue. The key property of this system is its additive homomorphism. In particular, given encryptions $E[b_0]$ and $E[b_1]$ of bits b_0 and b_1 respectively, the Proxy can non-interactively compute $E[b_0 \oplus b_1]$ as $E[b_0]E[b_1]$. Composition of commitments in this manner enables the Proxy to obtain an efficiently checkable proof of correct decommitment from the Sender, as we shall see. We sometimes refer to a Goldwasser-Micali ciphertext as a *quadratic-residue commitment*, abbreviated QR-commitment. We

use the very loose notation $E[b]$ to denote a Goldwasser-Micali encryption of (QR-commitment to) a bit b.

1.3 Other Work on Auctions

Because of the difficulties involved in deploying standard general secure function evaluation techniques, a number of other secure protocols have been proposed that are specially tailored for auctions. One of the earliest of these is the scheme of Franklin and Reiter [12]. This scheme is not fully private, in the sense that it only ensures the confidentiality of bids until the end of the protocol (although the authors mention a fully private variant). Some more recent schemes include those of Harkavy, Tygar, and Kikuchi [17], Cachin [5], Stubblebine and Syverson [25], Sako [24], Di Crescenzo [11], and Jakobsson and Juels [19]. The Harkavy *et al.* scheme is fully privacy preserving, but involves intensive bidder involvement [17], and is not easily adaptable to different auction types or to related protocols. The scheme of Cachin involves two servers, and requires some communication among bidders. At the end of the protocol, a list of bidders is obtained, but not the bid amounts. The scheme of Di Crescenzo [11] requires no communication between bidders, and has low round complexity, but involves the participation of only a single server. The scheme of Sako [24] works on a different principle from these others, involving opening of bids in what is effectively a privacy-preserving Dutch-style auction. Efficient for small auctions, it involves costs linear in the range of possible bids, and does not accommodate second-price and other auction types. The Jakobsson and Juels [19] protocol aims at streamlining general secure multi-party computation for functions that involve intensive bitwise manipulation, of which auction protocols, as mentioned above, are a good example. A very recent protocol is that of Baudron and Stern [1]. This protocol is expensive, and involves only a single server, with privacy ensured under the condition that there is no collusion between the auction server and any bidder.

Organization

Section 2 reviews some cryptographic building blocks required for our construction. In section 3, we consider some new methods for combining bit commitment with oblivious transfer, and introduce our VPOT protocol. We show how to apply VPOT to the problem of secure function evaluation in section 4. In section 5, we discuss the motivating example: private auction computations. Due to space constraints in this version of the paper, we do not provide formal security modelling.

2 Building Blocks and Background

We review several standard building blocks for our protocols. Further details regarding these primitives may found in the literature. We let $\in_U$ denote uniform,

random selection from a set. A useful summary of details of the Yao construction may be found in [20].

Private channels: We assume the use of private, authenticated channels between all three possible pairings of the Chooser, Proxy, and Sender. The private channel between the Chooser and Sender involves the Proxy as an intermediary, for the sake of protocol simplification. We assume that messages are authenticated in a non-repudiable fashion. We do not devote attention to the cryptographic elements underlying these channels. In practice, private channels may be realized by way of, e.g., the Secure Sockets Layer protocol (SSL) with supplementary use of digital signatures.

RSA-based 1-2 OT: Recall from above that the aim of 1-2 OT is for the Chooser to obtain a tag t_b for $b \in \{0,1\}$ from the Sender, who possesses the pair of tags (t_0, t_1). The Chooser should not learn t_{1-b}, and the Sender should not learn b. Most of the proposed practical 1-2 OT protocols in the literature rely on use of El Gamal encryption or some close variant. As an example, we describe the proxy oblivious-transfer protocol of NPS in detail at the beginning of section 3.

In this paper, we introduce a special, RSA-based 1-2 OT protocol. We do not make direct use of the RSA cryptosystem as such in the construction of this primitive. We do, however, employ the familiar RSA setup [23], which we briefly review here. An RSA public key consists of an RSA modulus $n = pq$, where p and q are primes, and a public exponent e such that $gcd(e, \phi(n)) = 1$. The corresponding private key d is such that $ed = 1 \bmod \phi(n)$. Our protocols involve exclusive knowledge and use of a private RSA key d by the Sender.

As a first step in the 1-2 OT protocol, the Sender must provide the Chooser with double commitments $C_0 = C_{k_0}(t_0)$ and $C_1 = C_{k_1}(t_1)$ on tags t_0 and t_1 respectively. The Sender additionally selects an integer $C \in_U Z_n^*$, which he sends to the Chooser. The Chooser, wishing to receive tag t_b, chooses an element $x \in_U Z_n^*$. If $b = 0$, the Chooser transmits $(x_0, x_1) = (x^3, Cx^3)$ to the Sender; otherwise, she transmits $(x_0, x_1) = (x^3/C, x^3)$. The Sender checks that $x_1/x_0 = C$. If so, he uses his private key to construct $(z_0, z_1) = (x_0^{1/3} k_0, x_1^{1/3} k_1)$, which he sends to the Chooser. The Chooser then makes use of x to extract k_b in the obvious fashion. Given k_b, the chooser can extract t_b from C_b as desired.

Lacking knowledge of the cube root C, the RSA assumption implies that the Chooser cannot obtain k_{1-b}. In the random oracle model, then, it may be seen that t_{1-b} is hidden from the Chooser in a semantically secure manner. As the Sender does not know for which element in the pair (x_0, x_1) the Chooser possesses the corresponding cube root, it may be seen that b is hidden in an information-theoretic sense from the Sender. Our VPOT protocol is essentially a variant on this basic 1-2 OT scheme.

As noted above, our choice of RSA for our protocols stems from a desire to render computation by the Chooser (corresponding to the bidder in an auction protocol) as efficient as possible. We employ $e = 3$, a common choice, in order to render these computations as rapid as possible, although none of our results depends on this fact.

Yao Circuit Evaluation: As discussed above, Yao circuit evaluation [26,20] serves as the cornerstone of our VPOT protocol, as it does for NPS. Informally, the Yao construction encrypts an entire boolean function, using ciphertexts to represent the 0 and 1's in a table composing a "boolean gate". It is easy to see how any function with finite domain and range can be compiled into a *circuit*, namely a finite set of interdependent boolean gates. Construction of Yao circuits is conceptually straightforward for auction functions, which incorporate a collection of '>' comparisons.

Goldwasser-Micali encryption: The concept of probabilistic encryption was introduced [16] and elaborated on in [4] to set forth the notion of semantic security in an encryption scheme. The basic scheme employs a Blum integer $n = pq$; this is the product of two primes, where each prime is congruent to 3 mod 4. (To facilitate our security proofs, we assume that the Blum integer employed here is not the same as the RSA modulus employed for 1-2 OT. In practice, and to simplify our protocol descriptions, we believe that use of the same modulus in both cases is acceptable.) The two primes constitute the private decryption key. Encryption is bitwise: a '0' bit is encoded as a random square modulo n, and a '1' bit as a random non-square modulo n with Jacobi symbol 1. In other words, the quadratic residuosity (QR) of a ciphertext indicates the value of the plaintext bit. Knowledge of p and q enables efficient determination of the quadratic residuosity of an element in Z_n.

The Goldwasser-Micali encryption scheme can be employed straightforwardly as a commitment scheme for a player that does not know the factorization of n. To decommit a commitment C_b as a '0' bit, a player provides a square root of C_b modulo n; to decommit as a '1' bit, she provides a square root of $-C_b$ modulo n. It is easy to see that the scheme is unconditionally binding. Privacy is reducible to the *quadratic residuosity assumption*. Recall from above that a key element of this encryption scheme, and indeed, our reason for employing it, is its useful additive homomorphism: $E[b_0]E[b_1] = E[b_0 \oplus b_1]$. We use this to prove the value of the XOR of two committed bits without revealing any additional information about the individual values of the bits themselves.

3 Verifiable Proxy Oblivious Transfer

As a basis for comparison, we begin by presenting details of the NPS proxy oblivious transfer protocol, whose intuition we sketched above. In an initialization process, the Chooser and Sender agree on a cyclic group G of order w over which computation of discrete logarithms is hard and an associated generator g, as well as a random value $C \in_U G$ whose discrete log is unknown to any player. As usual, we let b denote the choice of the bidder, the pair (t_0, t_1), the tags held by the Sender. The protocol is as follows [20]:

1. The Chooser selects a private key $x \in Z_w$, and computes a pair $(PK_b, PK_{1-b}) = (g^x, C/g^x)$, and sends PK_0 to the Sender via the Proxy. She sends x to the Proxy.

2. The Sender computes $PK_1 = C/PK_0$. The Sender computes the pair $(z_0, z_1) = (E_{PK_0}[\rho(t_0)], E_{PK_1}[\rho(t_1)])$, where E_{PK_i} denotes El Gamal encryption under public key PK_i and ρ denotes a suitable error-detection function. The Sender transmits the pair (z_0, z_1) to the Proxy in a random order.
3. The Proxy attempts to decrypt both values in the pair using x. The Proxy knows he has obtained t_b when the error-detection function ρ shows that the decryption is successful.

It may be observed that provided there is no collusion, neither the Sender nor the Proxy can learn b. It may be shown that under the Decisional Diffie-Hellman assumption, even if the Proxy and Chooser collude, they cannot learn both t_0 and t_1. The weakness in this protocol, hinted at above, is the fact that the Sender can choose to send $t_{b'}$ for b' of its choice simply by transmitting $(z_0, z_1) = (E_{PK_0}[\rho(t_{b'})], E_{PK_1}[\rho(t_{b'})])$. Even in the NPS auction protocol, neither the Chooser (i.e., a bidder) nor the Proxy (i.e., the auctioneer) can detect this tampering, which permits arbitrary alteration of bids.

We are now ready to remedy this problem by introducing our VPOT protocol. Due to space constraints, we provide only informal descriptions of the security properties of the protocol.

3.1 Verifiable Proxy Oblivious Transfer (VPOT)

VPOT detects the sort of cheating possible in NPS though use of a zero-knowledge proof based on QR-commitment. Here is a summary: the Sender provides a pair $(\mathcal{C}_0, \mathcal{C}_1)$ of commitments to the tags t_0 and t_1, in a random order. The Sender also commits to an ordering a of these commitments. In particular, $a = 0$ if $\mathcal{C}_0$ represents a commitment to t_0 (and $\mathcal{C}_1$ represents a commitment of t_1); otherwise, $a = 1$. The Sender provides this bit a for the Proxy as a QR-commitment of the form $E[a]$. The Sender obtains a share b_S of b, the ciphertext $E[b_S]$ being observed by the Proxy. The Proxy therefore can compute a commitment to the bit $c = a \oplus b_S$; in particular, $E[c] = E[a]E[b_S]$. If the Sender decommits correctly, the value c will specify whether the Proxy should be able to open $\mathcal{C}_0$ or $\mathcal{C}_1$. In particular, if $c = 0$, the Proxy should be able to open $\mathcal{C}_0$; otherwise the Proxy should be able to open $\mathcal{C}_1$. To prove correct behavior, the Sender decommits c for the Proxy by proving the quadratic residuosity of $E[c]$. Observe that the bit c, since it is "masked" by the secret bit a, does not reveal information about b_S to the Proxy. Hence the Proxy does not learn the bit b specifying the tag requested by the Chooser. The following is the full VPOT protocol.

1. The Sender chooses his desired tags $t_0, t_1 \in_U \{0,1\}^l$ and also an integer $C \in_U Z_n^*$.
2. The Sender computes commitments $\mathcal{C}_0 = \mathcal{C}_{k_a}(t_a)$ and $\mathcal{C}_1 = \mathcal{C}_{k_{1-a}}(t_{1-a})$. Let $u = E[a]$, i.e., a QR-commitment of a. The Sender also computes a commitment $CO = H[u]$ to ordering of $(\mathcal{C}_0, \mathcal{C}_1)$. The Sender transmits the pair $(\mathcal{C}_0, \mathcal{C}_1)$ to the Proxy, along with CO.

3. The Chooser receives C from the Proxy and splits b uniformly at random into bits b_P and b_S such that $b = b_P \oplus b_S$. She also selects $x \in_U Z_n^*$. If $b_P = 0$ she computes $x_0 = x^3$; otherwise, she computes $x_0 = x^3/C$. She also computes $v = E[b_S]$.
4. The Chooser sends (x_0, v, x) to the Proxy. She also sends (x_0, v) to the Sender (via the Proxy, if desired).
5. The Sender receives x_0 and computes $x_1 = Cx_0$. He then computes $y_0 = x_0^{1/3}$ and $y_1 = x_1^{1/3}$. He decrypts b_S. If $b_S = 0$, the Sender transmits the pair $(z_0, z_1) = (y_0 k_0, y_1 k_1)$ to the Proxy; otherwise he transmits the pair $(y_0 k_1, y_1 k_0)$.
6. The Sender transmits u to the Proxy (undoing the outer commitment in CO). The Sender then reveals $c = a \oplus b_S$ by decommitting $uv = E[a]E[b_S] = E[c]$. The decommitment of uv is provided as a value ρ such that $\rho^2 = uv$ if $c = 0$ and $\rho^2 = -uv$ if $c = 1$. The Proxy checks the correctness of these decommitments.
7. The Proxy first computes the cube of both z_0 and z_1 and checks that $H(z_0^3/x_0)$ and $H(z_1^3/x_1)$ are equal to the first element of $\mathcal{C}_0$ and the first element of $\mathcal{C}_1$, in either order. As a final step, the Proxy checks that he can use x to open $\mathcal{C}_0$ if $c = 0$ and $\mathcal{C}_1$ if $c = 1$. This check ensures that the Sender decommitted in the correct order.

Security Features:

- The Sender learns no information about b_P. Under the quadratic residuosity assumption governing the security of E, the Proxy does not learn b_S. Thus the Sender or Proxy cannot individually determine the Chooser's choice b.
- The Proxy cannot feasibly compute the cube root of C under the RSA assumption, and therefore cannot learn the cube roots of both x_0 and x_1. The unselected tag is in fact hidden in a semantically secure sense from the Proxy. This is true even if the Proxy cheats or colludes with the Chooser.
- The Proxy can verify that the Sender has correctly transmitted both t_0 and t_1, even though he can extract only one of them.
- The Proxy can verify that the Sender has correctly sent him t_b for the bit b selected by the Chooser. Assuming that the Sender and Proxy do not collude, therefore, the Chooser can be assured that the Proxy has received the correct tag t_b.

4 Two-Server Secure-Function Evaluation

In this section we describe an architecture for secure computation based on Yao circuits and VPOT. Due to lack of space, we cannot provide security modeling and proofs for our auction protocol in this paper. As above, we do assume the availability of private, authenticated channels among participating players.

4.1 Putting together VPOT and Yao Circuits

We now combine the VPOT protocol with Yao circuit to construct a secure function evaluation protocol involving two servers (evaluators) and multiple contributors of input values. For consistency with our protocol descriptions above, we refer to the two servers as the Proxy and the Sender. Our secure-computation protocol is designed to evaluate functions on inputs contributed by an arbitrarily large number m of players. We refer to these players as Choosers.

Our aim is to evaluate a function F on the m inputs of the Choosers. The Proxy and Sender together evaluate and publish the result of the function computation, and also provide each player with a receipt to guarantee correctness. The role of the Sender here is to construct Yao circuits and that of the Proxy, to evaluate these circuits. To compute input tags for the Yao circuits, these servers must process separate, parallel invocations of VPOT for every individual bit. The complete function evaluation protocol is as follows:

Offline Steps

1. The Sender generates an RSA modulus n and publishes this for the VPOT invocations in the current function evaluation session. (**Note**: It is in fact critical that a new RSA modulus be published for each session so as to ensure the privacy properties of VPOT across sessions.)
2. The Sender constructs N copies of Yao circuits to evaluate the function F. He sends these circuits to the Proxy, with double commitments to the garbled input tags, as in VPOT. He also publishes a lookup hash table enabling Yao-output-to-plaintext translation.
3. The Proxy selects half of the circuits at random, and asks the Sender to "open" them.
4. The Sender "opens" the selected circuits and sends the keys to all of their committed garbled input tags. This enables verification of their correct construction. This constitutes half of a cut-and-choose proof, the remaining half involving verification of consistent output on the unopened circuits.
5. The Proxy verifies that the "opened" circuits and committed input tags do indeed calculate the correct function.

VPOT steps

1. The Choosers submit their inputs bitwise to the Proxy according to the VPOT protocol.
2. The Proxy forwards these choices to the Sender according to the VPOT protocol.
3. The Sender sends the garbled input tags according to VPOT for each input bit, and also each of the $N/2$ unopened circuits.
4. If either the Proxy or Sender detects the presence of an ill-formed input by a Chooser, this is proven to the other server. Together the two servers can annul the input of any Chooser, provided that F is suitably constructed. Details are straightforward, and omitted here.

Circuit Evaluation

1. The Proxy checks the garbled tags against the commitments, and evaluates the unopened $N/2$ Yao circuits.
2. The Proxy looks up the Yao circuit outputs in the lookup tables, and verifies that the results of the $N/2$ trials are identical.
3. The Proxy publishes the output entries of the Yao tables, along with the function output. If the entries and output are correct, then the Sender certifies the output.

We remark that the Proxy should not publish the garbled output strings if the outputs are inconsistent. Such a situation only occurs if the Sender cheats, and revealing the outputs might leak information about input values. Once the correct output values are published, the result is verifiable by any outside party.

5 Application to Auctions

The two-server secure-function evaluation scheme presented in the previous section can be applied straightforwardly, of course, to create a sealed-bid auction system. As auctions are our key motivating application for the work in this paper, it is worth a brief, concluding discussion of the particular benefits of our approach to auctions.

As explained above, our scheme in this paper addresses the flaw in the NPS auction protocol [20]. The NPS protocol is privacy preserving, but effectively operates (unbeknownst to the authors) under the assumption that both the servers are honest. The flaw in this paper is simple: the sender may flip or set constant the two tags which he sends in the oblivious transfer for a given bit, e.g., he can send only '0' tags for a given bit submitted by a bidder. This allows the Sender to change the bid of any bidder to any value that the Sender chooses. Nonetheless, we believe that NPS offer a key insight in suggesting a two-server model to exploit the high computational efficiency and low round complexity of the Yao construction. This represents an important step toward the realization of practical, privacy-preserving auction protocols.

The secure-function evaluation procedure that we propose in section 4 not only fixes the flaw in the NPS protocol but, as already noted, has the additional benefit of substantially reducing the computation required by bidders. In summary, then, our proposed architecture offers the following features:

1. *Non-interactivity:* Bidders submit bids in a non-interactive fashion. That is, they present their bids to the servers, but need not participate subsequently in the auction protocol except to learn the outcome.
2. *Auction adaptability:* Our auction protocol is readily adaptable with little overhead to a range of auction types, such as highest-price auctions and Vickrey auctions.
3. *Full privacy:* We characterize privacy in terms of a static, active adversary that controls at most one server and an arbitrary number of bidders. The

only information revealed to such an adversary at the conclusion of the protocol about the bids of any honest bidders is the outcome of the auction. In a highest-price auction, for example, such an adversary learns only the winning bid and the identity of the winning bidder.

4. *Correctness:* Any player can be assured of the correctness of the auction execution assuming that the two auction servers do not collude.
5. *Robustness:* While we do not achieve robustness against failure by either of the auction servers, the servers can eliminate any ill-formed bids and process the remaining ones correctly.
6. *Low round-complexity:* The protocol involves only five communication passes; this includes the offline cut-and-choose proof of correct Yao circuit construction.
7. *High computational efficiency:* Our protocol is highly efficient in terms of computational requirements. For bidders, it is more so than any other cryptographically based privacy-preserving auction scheme in the literature. The requirement for a bidder in a typical auction would be several tens of modular multiplications (as opposed to a comparable number of modular *exponentiations* in NPS). The cost for the servers is about twice that in NPS. (While in general it is desirable to shed server load in favor of computation on the part of clients, the NPS protocol is so computationally intensive for clients as to pose a likely bottleneck even for reasonably powerful handheld devices.)

The principal drawback of our scheme is that, like the NPS protocol, it does not extend to a trust model involving more than two servers. Whether or not the NPS scheme can incorporate multiple servers is an open research question.

We emphasize that given the successful implementation experiments of NPS, our proposed architecture is likely to be amenable to practical software deployment. With this in mind, we provide a brief efficiency analysis in the appendix.

References

1. O. Baudron and J. Stern. Non-interactive private auctions. In S. Haber, editor, *Financial Cryptography '01*, pages 303–313, 2001.
2. D. Beaver. Minimal-latency secure function evaluation. In B. Preneel, editor, *Advances in Cryptology - Eurocrypt '00*, pages 335–350. Springer-Verlag, 2000. LNCS no. 1807.
3. M. Bellare, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *ACM CCS '90*, pages 503–513. ACM Press, 1990.
4. M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G.R Blakely and D. Chaum, editors, *Advances in Cryptology - Crypto '84*, pages 289–299. Springer-Verlag, 1985. LNCS No. 196.
5. C. Cachin. Efficient private bidding and auctions with an oblivious third party. In G. Tsudik, editor, *ACM CCS '99*, pages 120–127. ACM Press, 1999.
6. C. Cachin and J. Camenisch. Optimistic fair secure computation. In M. Bellare, editor, *Advances in Cryptology - Crypto '00*, pages 94–112. Springer-Verlag, 2000. LNCS no. 1880.

7. C. Cachin, J. Camenisch, J. Kilian, and J. Muller. One-round secure computation and secure autonomous mobile agents, 2000.
8. R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *Advances in Cryptology - Eurocrypt '01*, pages 280–300. Springer-Verlag, 2001. LNCS no. 2045.
9. Claude Crepéau. Verifiable disclosure of secrets and applications. In J.J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology - Eurocrypt '89*, pages 181–191. Springer-Verlag, 1990. LNCS no. 434.
10. Claude Crepéau, van de Graaf, Jeroen, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In D. Coppersmith, editor, *Advances in Cryptology - Crypto '95*, pages 110–123. Springer-Verlag, 1995. LNCS No. 963.
11. G. Di Crescenzo. Private selective payment protocols. In P. Syverson, editor, *Financial Cryptography '00*, 2000.
12. M. Franklin and M. Reiter. The design and implementation of a secure auction server. *IEEE Transactions on Information Theory*, 22(5):302–312, 1996.
13. M. Franklin and M. Yung. Varieties of secure distributed computing. In *Proc. Sequences II, Methods in Communications, Security and Computer Science*, pages 392–417, 1991.
14. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87*, pages 218–229. ACM Press, 1987.
16. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comp. Sys. Sci*, 28(1):270–299, 1984.
17. M. Harkavy, J.D. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *3rd USENIX Workshop on Electronic Commerce*, pages 61–73, 1999.
18. M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In T. Okamoto, editor, *Advances in Cryptology - Asiacrypt '00*, pages 143–161. Springer-Verlag, 2000. LNCS No. 1976.
19. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, editor, *Advances in Cryptology - Asiacrypt '00*, pages 162–177. Springer-Verlag, 2000. LNCS No. 1976.
20. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *1st ACM Conf. on Electronic Commerce*, pages 129–139. ACM Press, 1999.
21. M. Rabin. How to exchange secrets by oblivious transfer, 1991. Tech. Memo TR-81 Aiken Computation Laboratory, Harvard University.
22. R. L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer, 1999.
23. R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1977.
24. K. Sako. An auction protocol which hides bids of losers. In H. Imai and Y. Zheng, editors, *PKC '00*, pages 422–432. Springer-Verlag, 2000. LNCS no. 1751.
25. Stuart G. Stubblebine and Paul F. Syverson. Fair on-line auctions without special trusted parties. In *Financial Cryptography*, pages 230–240, 1999.
26. A.C. Yao. Protocols for secure computations (extended abstract). In *FOCS '82*, pages 160–164. IEEE Computer Society, 1982.

A Efficiency Considerations

The protocol VPOT involves both offline and online calculations; the latter may be considered of more practical relevance. A typical implementation might use a 1024-bit RSA modulus with exponent 3. Disregarding the cost of hash functions computations, which is relatively small, we observe that the Sender must compute seven modular multiplications offline. Online, the Sender must calculate three modular exponentiations. The Proxy has much less computational expense: only five modular multiplications and two modular divisions. Best of all, the Chooser need only calculate five modular multiplications per bit selection. Note that these are the costs for only one invocation of VPOT. A full auction protocol will involve many, of course, as we now consider.

A.1 A Typical Auction

To provide a flavor of the resource requirements for our proposed architecture, we summarize the computational requirements in a typical auction setting. We omit the relatively small cost of private channel establishment (via, e.g., SSL) and negligible cost of hash calculations; we count circuit evaluations as a unit.

In our example there are 10 bidders in the auction, the bids are 10 bits long, and 10 circuits out of 20 remain after the cut-and-choose step. The Sender must create the 20 circuits offline, and he can also calculate 10,000 of his modular multiplications off-line. During the protocol, he must calculate 2000 modular multiplications and 2000 modular exponentiations. The Proxy must evaluate 20 circuits accepting 100 inputs each, calculate 10,000 modular multiplications, and 2000 modular divisions. About half of this effort can be done off-line before bidding commences. Finally, the Choosers (bidders) need only perform at most 50 modular multiplications each in total to construct their bids.

Secure Vickrey Auctions without Threshold Trust

Helger Lipmaa[1], N. Asokan[2], and Valtteri Niemi[2]

[1] Laboratory for Theoretical Computer Science
Department of Computer Science and Engineering
Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Espoo, Finland
helger@tcs.hut.fi
[2] Nokia Research Center
P.O.Box 407, FIN-00045 NOKIA GROUP, Finland
{n.asokan,valtteri.niemi}@nokia.com

Abstract. We argue that threshold trust is not an option in most of the real-life electronic auctions. We then propose two new cryptographic Vickrey auction schemes that involve, apart from the bidders and the seller S, an auction authority A so that unless S and A collude the outcome of auctions will be correct, and moreover, S will not get any information about the bids, while A will learn bid statistics. Further extensions make it possible to decrease damage that colluding S and A can do, and to construct $(m+1)$st price auction schemes. The communication complexity between the S and A in medium-size auctions is at least one order of magnitude less than in the Naor-Pinkas-Sumner scheme.

Keywords: cryptographic auction schemes, homomorphic encryption, range proofs, Vickrey auctions.

1 Introduction

Vickrey auctions [Vic61] are sealed-bid auctions where the highest bidder is awarded the item, but is required to pay only the second-highest bid. Despite attractive theoretical properties, Vickrey auctions are relatively rarely used in practice since a cheating seller could either (1) change the outcome of auctions or (2) reveal bidders' private information. As argued in [RTK90,RH95], in the first case, a honest bid taker will not choose a Vickrey auction, while in the second case, a cheating bid taker eventually destroys the trust on which the use of Vickrey auctions depends. Therefore, Vickrey auctions are certainly more widely applicable when secured cryptographically, so that the seller is forced to follow the auction mechanism and no extra information is revealed to him. Attractive properties of Vickrey auctions together with these observations have motivated a huge body of research on cryptographic Vickrey auction schemes, starting with [NS93].

Now, most of the cryptographic auction schemes work in one of the following two trust models: (1) The *threshold (symmetric) trust model* where the tasks of the seller are executed by $N > 1$ servers, with less than $N/3$ (or $N/2$, depending on the precise cryptographic model) servers assumed to be dishonest; and (2) The *two-party (asymmetric) model* with a seller S and an auction authority A, where at least one of S and

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 87–101, 2003.

A is assumed to be honest. In this model, S and A are usually assigned complementary duties and are supposed to verify the actions of each other.

We argue that the threshold trust model is not suitable for many real-life electronic auctions. This trust model requires that each replicated server should be run by an independent auction authority, of which a majority is more trustworthy than the seller. The number of such trusted authorities is likely to be rather small compared to the number of distinct sellers. Therefore, every auction authority will participate in many auctions conducted by many different sellers. But in each auction, this authority has to do the same amount of work as the sellers. That may quickly lead to the auction authorities becoming bottlenecks either in the sense of security or efficiency. Simplistic use of the threshold trust approach is therefore not scalable in the case of applications like electronic auctions. (See also [NPS99] for additional motivation.)

In this paper we propose two different schemes that work in the two-party model. The first scheme is intended to illustrate the basic properties of this model. In this scheme, S blindly shuffles encrypted bids before forwarding them to A. After that, A computes the second highest bid X_2 and sends it together with a pointer to the winner's encrypted bid to S. The seller S then identifies the winner. At the end, any bidder can complain if he believes the result to be incorrect. In particular, if all bidders confirm the linear order between their bid b and X_2 (i.e., whether $b < X_2$, $b = X_2$ or $b > X_2$), A becomes accountable for his actions. However, this simple scheme has several vulnerabilities that we outline later.

The main contribution of this paper is *the homomorphic auction scheme*. In this scheme, a bid b is encoded as B^b, B being the (maximum allowed) number of bidders. The ith bidder encrypts his bid b_i with A's public key in a suitable homomorphic encryption scheme, and sends it to S. S multiplies all the received encrypted bids, and sends the resulting encryption $B^{\sum_i b_i}$ to A. After decrypting this result, A finds out the bid statistics (that is, how many bidders bid b for any possible bid b) but is not able to connect any bidders with their bids. Then, A sends the second highest bid to S. Every action in this scheme is accompanied with an *efficient* (statistical) zero-knowledge correctness proof. By using recently proposed cryptographic range proofs, we achieve that both the bidder-seller and the seller-authority communication complexity are of order $\Theta(V \cdot \log_2 B)$ bits, where V is the maximum possible number of different bids. In medium-size auctions this results in amount of interaction, at least one order of magnitude less than in the Naor-Pinkas-Sumner scheme [NPS99] that was the only previously known secure Vickrey auction scheme without threshold trust.

Our schemes use a few cryptographic observations that might be of independent interest. First, the homomorphic scheme uses a property of some known homomorphic public-key cryptosystems that we call *coin-extractability*; the same property might be also important in other applications. (In the context of auction schemes, coin-extractability of the Paillier cryptosystem was already used in [BS01].) We propose a range proof in exponents that corrects a few mistakes in the proof system from [DJ01, Section 5]; it seems to be the most efficient currently known scheme with perfect zero-knowledge that works with arbitrary intervals; a more efficient statistical zero-knowledge proof system that works only with a prime base was recently proposed in [Lip01]. Based on either of these proof systems and a recent range proof from [Lip01] we describe an efficient

noninteractive statistical zero-knowledge proof system for proving that an encrypted value is the second highest value in some set. This proof system is used by A to prove that he computed a correct second-highest bid X_2 and can be easily extended to prove that an encrypted value is the $(m+1)$st highest value for a small $m > 1$. This results, in particular, in efficient $(m+1)$st price auctions.

Road-map. We start with a short overview of the existing auction schemes in Section 2. We give the necessary (cryptographic) preliminaries for the rest of the paper in Section 3. In Section 4, we describe several auxiliary protocols for homomorphic public-key cryptosystems. Our new auction schemes are described in Section 5, some extensions to them are given in Section 6, followed by some discussion in Section 7. We compare our schemes with the Naor-Pinkas-Sumner scheme in 8, and conclude the paper in Section 9.

2 State of the Art

We will briefly survey the known cryptographic Vickrey auction schemes that do not rely on the threshold trust. A few auction schemes [Cac99,BS01] are based on the Yao's millionaire's problem. Such schemes avoid threshold trust by using an oblivious third party for bid comparison. Without a collusion between the seller and the third party, the seller will get to know some partial order among the bids but not the bid values themselves. While [BS01] also discusses how to extend their auction scheme to the Vickrey auctions, at least the extension proposed in their paper would also reveal the identity of the second highest bidder. This together with the partial leak of information to the untrusted seller poses a serious problem, since it demotivates potential high bidders to participate in that type of an auction.

The auction scheme of Naor, Pinkas and Sumner [NPS99] uses a third party A (that we call *an auction authority*) and no unnecessary information is leaked unless the seller S and the third party A collude. The Naor-Pinkas-Sumner scheme bases on the two-party secure computation model of Yao, where A constructs a garbled circuit, transports it (off-line) to S and then helps S (on-line) to execute it. The circuit can be designed to completely satisfy all possible security requirements. However, A may "misdesign" the circuit to perform whatever computations he likes. A serious drawback of this scheme is that a corrupt third party can only be detected by "cut-and-choose" techniques [NPS99, Section 2.4] that would introduce a severe overhead to the protocol. The authors suggest that A (called an auction issuer in their scheme) should be an established service provider with high reputation, while S is a (usually considerably less trusted) seller. They argue that even cheating once would ruin the reputation of A.

On the other hand, even if the cut-and-choose technique is not used, circuit transfer would imply a huge communication complexity between A and S. Even if done off-line, the amount of information transferred is clearly infeasible in many real-life scenarios. Another drawback is that the circuit depends on the maximum number of bidders and hence the seller has to estimate this number before the auction relatively precisely.

Currently, [NPS99] seems to be the only published secure Vickrey auction scheme that neither reveals any unnecessary information nor relies on the threshold trust. Moreover, we are aware of only one other secure Vickrey auction scheme, recently proposed

by Kikuchi [Kik01]. Kikuchi's scheme has smaller communication complexity than the Naor-Pinkas-Sumner scheme but relies on threshold trust. Moreover, the number of bidders in Kikuchi's scheme is bounded above by the number of auction servers, which makes it unusable in many practical situations. (However, it is still applicable, for example, in radio frequency spectrum or wireless spectrum license auctions, where the number of competitors is relatively small.)

The Sakurai-Miyazaki auction scheme [SM00] is secure without an *explicit* threshold-trust assumption. However, this scheme uses a bulletin board, a secure implementation of which introduces implicit threshold trust. It also bases on some relatively ad hoc security primitives. Finally, there are also schemes where threshold trust is w.r.t. the bidders. However, in these schemes, the threshold trust assumption seems to have even less ground, since in many practical cases, there is no guarantee that even a single bidder will be honest.

3 Preliminaries

Notation. Let B be the (maximum) number of bidders, let V be the (maximum) number of different bids. After an auction, let $(X_1, \ldots, X_B)$ be the vector of bids in a nonincreasing order, and let Y_j be the bidder who bid X_j. Let t denote the security parameter. For a probabilistic public-key cryptosystem (G, E, D), let $c = E_K(m; r)$ denote the encryption of m by using a random coin r under the key K. In general, we denote the message space by $\mathcal{M}$, the key space by $\mathcal{K}$, the nonce space by $\mathcal{R}$ and the ciphertext space by $\mathcal{C}$.

Homomorphic encryption. Let G be the key generation algorithm, E the encryption algorithm and D the decryption algorithm. We say that a public-key cryptosystem $\Pi = (G, E, D)$ is *doubly homomorphic* if the sets $\mathcal{M}$ and $\mathcal{R}$ are (additive) Abelian groups, and $E_K(m_1; r_1) \cdot E_K(m_2; r_2) = E_K(m_1 + m_2; r_1 + r_2)$ for every $(K, m_1, m_2, r_1, r_2) \in \mathcal{K} \times \mathcal{M}^2 \times \mathcal{R}^2$. If Π is doubly homomorphic then $E_K(em; er) = E_K(m; r)^e$ for all e, and $E_K(m; r) = E_K(0; r) \cdot E_K(m; 0)$. In most of the known (doubly) homomorphic public-key cryptosystems, all spaces $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{C}$ are key-dependent: In such cases we assume that the corresponding key K is understood from the context. With this in mind, we denote $M := \lceil \log_2 |\mathcal{M}| \rceil$, $R := \lceil \log_2 |\mathcal{R}| \rceil$, $C := \lceil \log_2 |\mathcal{C}| \rceil$.

Damgård-Jurik cryptosystem [DJ01]. Damgård-Jurik cryptosystem is an extension of the Paillier cryptosystem with the main difference that the size of message space can be increased polynomially without increasing $|\mathcal{R}|$ at the same time. Here, $K = n$ is an RSA modulus and s is a public parameter. The message space $\mathcal{M} = \mathbb{Z}_{n^s}$, coin space $\mathcal{R} = \mathbb{Z}^*_{n^{s+1}}$ and ciphertext space $\mathcal{C} = \mathbb{Z}^*_{n^{s+1}}$ vary together with the key n. In one variant of this cryptosystem, a message m is encrypted by generating a random number r and letting $E_K(m; r) := (1 + n)^m \cdot r^{n^s} \mod n^{s+1}$.

Coin-extractability. We say that (G, E, D, R) is a *coin-extractable public-key cryptosystem* if (G, E, D) is a public-key cryptosystem and R is an efficient algorithm, such that $R_K(E_K(m; r)) = r$ for all m and r. The Damgård-Jurik cryptosystem is coin-extractable since after decrypting $c = E_K(m; r)$, receiver obtains $r^{n^s} \mod n^{s+1}$.

Since he knows factorization of n, he can then easily find r. Coin-extrability of the Paillier cryptosystem was also used in [BS01]. Note that we let $\mathcal{R}$ to be an additive group even if in the Damgård-Jurik cryptosystem, $\mathcal{R} = \mathbb{Z}^*_{n^{s+1}}$ is a multiplicative group.

Proofs of knowledge. For some (unknown) bit-string α and predicate $P(\cdot)$, $\mathrm{PK}(y = P(\alpha))$ is a (usually, honest-verifier zero-knowledge) proof-of-knowledge between two parties that given the publicly known value y, the first party knows a value of α, such that the predicate $P(\alpha)$ is true. The convention is that Greek letters denote the knowledge proved, whereas all other parameters are known to the verifier. We assume that the "Greek variables" are scoped within one proof-of-knowledge. For example, $\mathrm{PK}(c = E_K(m;\rho))$ is a proof that given as common input a ciphertext c, plaintext m and a public key K, the prover knows a nonce ρ, such that $c = E_K(m;\rho)$. For many predicates, the corresponding proofs are already known; for a few predicates we will devise new proofs in Section 4.

In our subsequent protocols, we will need two proofs-of-knowledge from [DJ01]. The first proof system is for $\mathrm{PK}(c = E_K(m_1;\rho) \vee c = E_K(m_2;\rho))$; we call this a *1-out-of-2 proof system.* A noninteractive version of this proof is $3t + 2R$ bits long. The second proof system is for $\mathrm{PK}(c_1 = E_K(\mu_1;\rho_1) \wedge c_2 = E_K(\mu_2;\rho_2) \wedge c_3 = E_K(\mu_3;\rho_3) \wedge \mu_1\mu_2 = \mu_3)$; we call this a *proof system for multiplicative relationship.* A noninteractive version of this proof is $2t + M + 2R$ bits long.

Range proof. In rest of this paper we will assume that $\mathcal{M}$ is a linearly ordered ring so that we could apply the recent range proof by Lipmaa [Lip01] for $\mathrm{PK}(c = E_K(\mu;\rho) \wedge \mu \in [\mathsf{L},\mathsf{H}])$. We will briefly outline this proof system. Prover and Verifier use both a doubly homomorphic public-key cryptosystem and an integer commitment scheme [DF01]. Now, $\mu \in [\mathsf{L},\mathsf{H}]$ can be proven by first showing that $\mu - \mathsf{L} \geq 0$ and then showing that $\mathsf{H} - \mu \geq 0$. Thus, it suffices to describe a proof system for $\mathrm{PK}(c = E_K(\mu;\rho) \wedge (\mu \geq 0))$ that proceeds as follows: (1) Prover commits to μ and proves in statistical zero-knowledge that the committed number is equal to $\mu \pmod{|\mathcal{M}|}$. (2) Prover finds a representation $\mu = \mu_1^2 + \mu_2^2 + \mu_3^2 + \mu_4^2$ of μ. (Such representation exists iff $\mu \geq 0$ as shown by Lagrange. An efficient algorithm for finding μ_i was proposed by Rabin and Shallit [RS86].) Prover commits to $(\mu_1,\mu_2,\mu_3,\mu_4)$ and then proves in statistical zero-knowledge that $\sum_{i=1}^4 \mu_i^2 = \mu$. With suitable security parameters, a noninteractive version of this proof is $\approx 3366 + \frac{5}{8}\lceil \log_2 \mathsf{H} \rceil$ bytes long. For more details see [Lip01].

4 Auxiliary Proofs

Range Proof in Exponents. In the following we will need a proof system for $\mathrm{PK}(c = E_K(B^\mu;\rho) \wedge (\mu \in [0,\mathsf{H}]))$; we call such a proof system a *range proof in exponents.* Our proof system is based on the observation that $\mu \in [0,\mathsf{H}]$ iff $\mu = \sum_{j=0}^{\lfloor \log_2 \mathsf{H} \rfloor} \mathsf{H}_j \cdot \mu_j$ for some $\mu_j \in \{0,1\}$ and $\mathsf{H}_j := \lfloor (\mathsf{H} + 2^j)/2^{j+1} \rfloor$. For example, $\mu \in [0,10]$ iff $\mu = 5\mu_0 + 3\mu_1 + \mu_2 + \mu_3$, whereas $\mu \in [0,9]$ iff $\mu = 5\mu_0 + 2\mu_1 + \mu_2 + \mu_3$. Equivalently, $\mu \in [0,\mathsf{H}]$ iff $B^\mu = \prod_{j=0}^{\lfloor \log_2 \mathsf{H} \rfloor} (B^{\mathsf{H}_j})^{\mu_j}$ for some $\mu_j \in \{0,1\}$. Based on this, we can prove that

Theorem 1. *Let (G, E, D) be the Damgård-Jurik cryptosystem; we follow our notational convention that $\mathcal{R}$ is an additive group. For $j \in [0, \lfloor \log_2 \mathsf{H} \rfloor]$, let $\mathsf{H}_j := \lfloor (\mathsf{H} + 2^{j-1})/2^j \rfloor$. Then the next protocol is a complete, HVZK and specially sound proof system for $\mathrm{PK}(c = E_K(B^\mu; \rho) \wedge \mu \in [0, \mathsf{H}])$. Let $c_{2,-1} \leftarrow E_K(1; 0)$. For all $j \in [0, \lfloor \log_2 \mathsf{H} \rfloor]$ do:*

- *Both Verifier and Prover precompute H_j. Prover generates an r_j s.t. $\sum_{j=0}^{\lfloor \log_2 \mathsf{H} \rfloor} r_j = \rho$, and a $c_{1j} \leftarrow E_K((B^{\mathsf{H}_j})^{\mu_j}; r_j)$. Prover sends c_{1j} to Verifier. Both parties compute $c_{2j} = \prod_{k=0}^{j} c_{1k}$. Prover proves to Verifier that c_{1j} is an encryption of either 1 or B^{H_j} by using a 1-out-of-2 proof system from [DJ01] and that $\mathrm{PK}(c_{2,j-1} = E_K(\mu_1; \rho_1) \wedge c_{1j} = E_K(\mu_2; \rho_2) \wedge c_{2j} = E_K(\mu_3; \rho_3) \wedge \mu_1 \mu_2 = \mu_3)$, by using a proof system for multiplicative relationship from [DJ01].*

Moreover, $\mathrm{PK}(c = E_K(B^\mu; \rho) \wedge \mu \in [\mathsf{L}, \mathsf{H}])$ can be proven similarily by taking $\mathsf{H}_j = \lfloor (\mathsf{H} - \mathsf{L} + 2^j)/2^{j+1} \rfloor$ and adding an extra addend $\mathsf{H}_{-1} = \mathsf{L}$. A proof for $\mathrm{PK}(c = E_K(B^\mu; \rho) \wedge \mu \geq \mathsf{L})$ can now be derived by letting $\mathsf{H} = |\mathcal{M}| - 1$.

Note that if $c = E_K(B^\mu; \rho)$ for $\mu \in [0, \mathsf{H}]$ then $(B^{\mathsf{H}_j})^{\mu_j} \in \{1, B^{\mathsf{H}_j}\}$. Completeness of this proof system follows since $c_{2j} \leftarrow E_K(\sum_{k=0}^{j} (B^{\mathsf{H}_k})^{\mu_k}; \sum_{k=1}^{j} r_k)$ (hence $c_{2,\lfloor \log_2 \mathsf{H} \rfloor} = c$), $D_K(c_{2j})/D_K(c_{2,j-1}) = D_K((B^{\mathsf{H}_j})^{\mu_j}) = D_K(c_{1j})$, and both protocols from [DJ01] are complete.

Let $\ell_1 = 3t + 2R$ be the length of the 1-out-of-2 proof system from [DJ01] and let $\ell_2 = 2t + M + 2R$ be the length of the proof system for multiplicative relationship from Lemma [DJ01]. Clearly, a noninteractive version of the protocol from Theorem 1 is then $(\lfloor \log_2 \mathsf{H} \rfloor + 1) \cdot (C + \ell_1 + \ell_2) = (\lfloor \log_2 \mathsf{H} \rfloor + 1) \cdot (C + 5t + 4R + M) \leq \log_2 V \cdot (C + 5t + 4R + M)$ bits long.

Damgård and Jurik presented a very similar range proof in exponents in [DJ01]. However, their proof system had a subtle flaw of working only when H is $2^j - 1$ for some j. The sole difference between our proof system and the one in [DJ01, Section 5] is in the choice of the values H_j: Namely, Damgård and Jurik chose H_j to be the j-th bit in the binary expansion of H, while we have chosen different H_j, so that values $\mu \in [0, \mathsf{H}]$ have at least one (but possibly several different) representations $\sum \mathsf{H}_j \mu_j$, but values from outside of this interval do not have such representations. Our protocol has identical complexity to the protocol from [DJ01] since the values H_j can be precomputed by both parties separately. We were recently acknowledged [Dam01] that the authors of [DJ01] were aware of the flaw in [DJ01] and have a different solution to it. However, their new protocol, described in the upcoming journal version [DJN], requires in particular approximately $2 \log_2 \mathsf{H}$ additional proof systems for multiplicative relationship. This means that compared to their solution we save a constant factor in the size of interactive protocol.

Proof That X_2 is Second Largest in Set. Let (G, E, D, R) be a coin-extractable doubly homomorphic public-key cryptosystem like the Damgård-Jurik cryptosystem. In Protocol 1, Prover P and Verifier V have a common input $(X_2, \mathsf{tiebreak}, c)$, where $D_P(c) = \sum_j x_j B^{b_j}$ for some $x_j \in [0, B-1]$. Prover has to prove to Verifier that (1) If $\mathsf{tiebreak} = 0$, then there is exactly one j_0, such that $b_{j_0} > X_2$, and exactly one j_1,

Protocol 1 Proof that $(X_2, \text{tiebreak})$ is correctly computed.

1. P finds $x \leftarrow D_P(c)$ and $r \leftarrow R_P(c)$. He decodes x in base B as $\sum_j x_j B^j$. Based on this, P finds $X_1 \leftarrow \max\{j : x_j > 0\}$ and X_2.
2. If tiebreak $= 0$ (there is no tie-break), then do:
   a) P proceeds as follows. Let $r_1 \leftarrow_R \mathcal{R}$, $c_1 \leftarrow E_P(B^{X_1}; r_1)$ and $c_2 \leftarrow E_P(x - B^{X_2} - B^{X_1}; r - r_1)$. Send (c_1, c_2) to V.
   b) V verifies that $c_1 \cdot E_P(B^{X_2}; 0) \cdot c_2 = c$.
   c) After that, P proves to V, that
      i. c_1 encrypts a $(> X_2)$th power of B: $\text{PK}(c_1 = E_P(B^\mu; \rho) \wedge \mu \in [X_2 + 1, V])$.
      ii. $\text{PK}(c_2 = E_P(\mu; \rho) \wedge \mu \in [0, (B-2) \cdot B^{X_2 - 2} - 1])$ by using the range proof.
3. Otherwise (if there is a tie-break), do:
   a) P sends $c_2 \leftarrow E_P(x - 2(B^{X_2}); r)$ to V.
   b) V verifies that $(E_P(B^{X_2}; 0))^2 \cdot c_2 = c$.
   c) P proves to V that c_2 encrypts a value less than $(B-2) \cdot B^{X_2}$: $\text{PK}(c_2 = E_P(\mu; \rho) \wedge \mu \in [0, (B-2) \cdot B^{X_2} - 1])$.

such that $b_{j_1} = X_2$, and (2) If tiebreak $= 1$, then there are no such j-s, for which $b_j > X_2$, but there exist $j_0 \neq j_1$, such that $b_{j_0} = b_{j_1} = X_2$. Let ℓ_1 be the length of the used range-proof-in-exponents, and ℓ_2 be the length of the used range proof. Then a noninteractive version of Protocol 1 is $\leq 2C + \ell_1 + \ell_2$ bits long.

5 New Auction Schemes

We now present our auction schemes. In the following schemes, all parties are assumed to have a public encryption key and a signature key that are in public knowledge. The signature scheme should be secure against the chosen-message attack. In the scheme of Section 5.2, A also has a public integer commitment key. We assume that the key-distribution mechanism is secure.

5.1 Simple Scheme

Protocol 2 depicts a simple auction scheme that puts more trust on A, compared to the later scheme from Section 5.2, but avoids elaborated cryptographic protocols and does not put as severe limites on the values B and V as the latter. If tiebreak $= 0$ (no tie-break), a successful protest constitutes bidder i_w proving (in zero-knowledge) that he did not bid more than X_2, or some other bidder proving that he bid also more than X_2. If tiebreak $= 1$, a successful protest means i_w proving that he bid less than X_2, or some other bidder proving that he bid more than X_2. All such proofs can be based on the previously described range proofs that originate from [Lip01].

In this auction scheme, A will get to know the winner and the bid statistics, but cannot bind bids with concrete bidders. A malicious A can change X_2 to X_2', $X_1 > X_2' \geq X_2$. The seller S will get to know only the minimal amount of information: That is, X_2, and the winner (or all winners if there is a tie-break). If S and A do not collude, the seller cannot deviate from the protocol without being detected.

Protocol 2 The simple auction scheme.

BIDDING PHASE

1. Bidder i encrypts b_i by using A's public key and sends the resulting ciphertext c_i together with $\mathrm{sig}_i(c_i)$ to S via a confidential channel.
2. S verifies the signatures (complains, if necessary). He computes $z \leftarrow \mathrm{sig}_S(\{c_i\})$, where $\{c_i\}$ is represented in some fixed order that does not depend on i-s. (For example, in lexicographic order with respect to the c_i-s.). He broadcasts $(\{c_i, \mathrm{sig}_S(c_i)\}, z)$ to all bidders.
3. Every bidder i obtains $(\{c_{i'}, \mathrm{sig}_S(c_{i'})\}, z)$, and complains if c_i is missing. He also verifies the signature z.

BID OPENING PHASE

1. A does the following. Obtain $\{(c_i, \mathrm{sig}_S(c_i))\}$ and z, and verify the signatures $\mathrm{sig}_S(c_i)$, $\forall i$, and z. For all i: Decrypt c_i and obtain b_i. Compute the second highest bid X_2. Set tiebreak $\leftarrow$ 1 if there is a tie-break and tiebreak $\leftarrow$ 0, otherwise. Send $(X_2, \mathsf{tiebreak})$, together with signature $z' \leftarrow \mathrm{sig}_A(X_2, \mathsf{tiebreak}, \{c_i\})$ to S.
2. S verifies the signature z'. He then broadcasts $(X_2, \mathsf{tiebreak}, z, z')$ to all bidders.
3. After obtaining $(X_2, \mathsf{tiebreak}, z, z')$, all bidders verify the signature z', in particular that it is given over the same set $\{c_i\}$ as z.
4. A points to S a c_{i_w}, such that $D_A(c_{i_w}) = X_1$. S identifies i_w and declares him the winner.
5. Bidders can now protest against the choice of i_w.

5.2 Homomorphic Scheme

Protocol 3 depicts *the homomorphic scheme*, where every bid b_i is encoded as B^{b_i}. This encoding will allow everybody to compute, given encryptions of B^{b_i}, an encryption of $\sum_i B^{b_i}$ without knowing the corresponding decryption key. Note that for this scheme to work correctly it is necessary that $B^V < |\mathcal{M}|$. We assume implicitly that communication goes over a confidential channel.

The auction authority will get to know the bid statistics but cannot bind them with the bidders. The seller will get to know only the minimal amount of information, X_2 and the winner (or all winners if there is a tie-break). If S and A do not collude, neither S nor A can deviate from the protocol without being detected.

In many situations, knowing bid statistics might not be very valuable for A: First, even if the authority does sell the statistics to the seller of a subsequent auction, the new seller will most probably not have exactly the same set of bidders. Second, if S would use designated verifier signatures (with verifier A), A would be unable to convince the new seller that he is actually selling correct data, unless carefully building up reputation of a "honest cheater". However, such a reputation is unlikely to stay hidden from bidders for extended periods of time. Third, even if it is impossible to verify for sure whether A abuses the bid statistics, but too obvious abuses will certainly be noticed and ruin his reputation.

The confirmation step is optional, since the proof of step 4 already shows that X_2 is correctly computed. The highest bidder has to participate in the confirmation phase to claim the item. However, if he does not, one can apply a mandatory protocol where every bidder has either to confirm or revoke that he is eligible to win.

Protocol 3 The homomorphic auction scheme.

BIDDING PHASE

1. Bidder i encodes and encrypts his bid b_i by using A's public key, $c_i = E_A(B^{b_i}; r_i)$, signs it, and sends $(c_i, \mathrm{sig}_i(c_i))$ to S. Bidder i proves to S that the bid is correctly computed by performing a proof for $\mathrm{PK}(c_i = E_A(B^{\mu}; \rho) \wedge (\mu < V + 1))$.
2. S does the following: Verify the signatures and complain if necessary. Let $c \leftarrow \prod_i c_i = E_A(\sum_i B^{b_i}; \sum_i r_i) = E_A(\sum_j x_j B^j; \sum_i r_i)$, $\mathcal{C} \leftarrow \{c_{\pi(i)}\}$ for random permutation π, $h \leftarrow H(\mathcal{C})$ and $z = \mathrm{sig}_S(h, c)$. Send $\mathcal{C}$ to all bidders. Post (c, z).
3. For all i, bidder i verifies that $c_i \in \mathcal{C}$, $c = \prod_{c' \in \mathcal{C}} c'$ and $z = \mathrm{sig}_S(h, \mathcal{C})$.

BID OPENING PHASE

1. A obtains (c, z) and verifies the signature z.
2. After that, A decrypts c, obtains $D_K(c) = \sum_j x_j B^j$ and then computes the second highest bid X_2 and a bit tiebreak, such that tiebreak $= 1$ iff there is a tie-break. He sends $(X_2, \mathsf{tiebreak})$, together with his signature $z = \mathrm{sig}_A(X_2, \mathsf{tiebreak})$, to S.
3. S verifies z.
4. A proves to S that the pair $(X_2, \mathsf{tiebreak})$ is correctly computed. (See Section 4 for the corresponding proof.)
5. S publishes X_2 on an authenticated medium together with A's and his own signatures.
6. Bidders can now participate in the confirmation phase with S. If tiebreak $= 0$, the bidder who confirms that he bid more than X_2 will be the winner. If tiebreak $= 1$, a previously announced rule (for example, the equal probability rule) is used to determine the winner.

6 Refinements to Our Auction Schemes

Using a prime B. If B is a prime, the range proofs in exponents can be made considerably shorter as shown in [Lip01]. Without going into more details, we note that a noninteractive version of this proof has length $2636 + \lceil \log_2 |\mathcal{M}| \rceil + \frac{5}{16} \log_2 \mathsf{H}$ bytes. Now, restricting B to be a prime is not a big obstacle in our auction scheme. Really, by the prime number theorem, the average gap $p_{i+1} - p_i$ between two consequent primes less than n is $\Theta(\log_2 n)$. In particular, the largest prime gap between primes less than 1200 is 22. Thus, the seller has to introduce approximately $\Theta(\log_2 B)$ dummy bidders that do not actually participate in the auction.

Extension to $(m+1)$st price auctions. Vickrey auction mechanism can be generalized to the $(m+1)$st price auction mechanism, where m copies of the same item are given to m highest bidders for the $(m+1)$st highest bid. The $(m+1)$st price auction scheme is a direct incentive-compatible mechanism [Vic61]. A trivial modification to our homomorphic auction scheme results in a $(m+1)$st price auction scheme with additional communication of about $(m-2) \cdot (C + \ell)$ bits, where $\ell \leq 2(C + 5t + 4R + M) \log_2 V$ is the length of the range proof in exponents from Section 4. Briefly, this modification consists of changing Protocol 1 so that instead of proving that $D_K(c) = B^{X_1} + B^{X_2} + \tau$ with $\chi_1 > X_2$ and $\tau < B^{X_2+1}$, P proves that $D_K(c) = \sum_{i=1}^{m} B^{\chi_i} + B^{X_2} + \tau$ with $\chi_i > X_2$ and $\tau < B^{X_2+1}$. On the other hand, if we assume that B is a prime then usually $\ell \leq 3$ KB. The only previous secure $(m+1)$st-price auction schemes that we

are aware of by Kikuchi [Kik01] and by Naor, Pinkas and Sumner [NPS99]. In the latter scheme, circuit for the mth price auctions is about $m \log_2 V$ times bigger than circuit for the first price auctions [Pin01].

Thresholding. It is possible to distribute A and/or S using the threshold trust model. For example, when trust on A is distributed in a way where bid statistics will only be leaked if at least $1/3$rd of the A-servers are faulty then in the homomorphic scheme the thresholded A does not have to prove that X_2 was correctly computed. This introduces a new interesting *bipartite threshold trust model*, where some of the functionality is controlled by one set $\mathcal{S}$ of servers (operated by one or more parties), while some other functionality is controlled by another set $\mathcal{A}$ of servers (operated by one or more parties, independent from the parties who operate the set $\mathcal{S}$). Server sets $\mathcal{S}$ and $\mathcal{A}$ check that the other set behaves correctly. Our auction schemes stay secure unless significant fractions of both $\mathcal{S}$ and $\mathcal{A}$ cheat. We feel that this bipartite threshold trust model might also be interesting in many other applications, like e-voting.

Reducing the influence of collusions. Let H be a secure commitment scheme; in practice one may assume that H is a hash function. Damage caused by colluding A and S can be reduced in both of our auction schemes when the bidders first send a signed commitment to their bid to S, who then broadcasts all commitments together with his signature on the tuple of commitments. Only after that, actual encrypted bids are sent to S.

When this "meta-scheme" is employed, an auction will stay correct even when S and A collude. The only use from the colluding is that then A and S will obtain additional information: Namely, they will be able to connect every bidder with his bid. However, they will *not* be able to artificially raise X_2 or declare a false winner. The same simple but very useful method works in conjunction with almost every auction (and voting) scheme.

There are three scenarios how this meta-scheme itself could be abused. First, a bidder can cheat by sending a commitment but then refusing to send the bid. However, since the commitments are signed the offending bidder is identifiable in some sense. Second, colluding S and A can delete some bids that are not to their liking, arguing that the corresponding encrypted bid was not submitted. However, in this case corresponding bidders can prove by showing their bids that S (and A) were faulty. Third, S and A can arrange a shill to submit a very low fake bid. If then the results are not to their liking, they can claim that the shill failed to send the encryption. However, this shill is again identifiable. Hence, this concern might not be very serious especially in the local electronic auctions. (A more general solution would be to use a fair exchange instead of receipts during the bid commitment.)

Avoiding replay attacks. Because of the reliance on homomorphism, encrypted bids cannot contain any other information but B^{b_i}. This opens up cut-and-paste attacks that may compromise bid privacy. As a simple example, a crook seller that wants to find the highest bid in an auction could replay the winning bid, along with a bunch of zero bids and one artificially high bid $b = V - 1$ to A.

Replay attacks can be avoided by once again using the coin-extractability property of the cryptosystem, as far as the random coins r_i are not revealed to the seller. Namely,

accompany each bid with $E_A(\mathsf{transaction\_id}; r_i)$, where r_i is the same coin that was used to encrypt the bid, and transaction_id is guaranteed to be unique (i.e, something that A can detect in case of a replay).

Preferably transaction_id should also contain a commitment of auction parameters. For example, transaction_id can be computed as $H(\mathsf{auction\_advertisement})$ where auction_advertisement includes all relevant details about the auction (e.g., seller name, sequence number added by seller, auction mechanism, deadlines, etc.). Since this is the only communication channel (for arbitrary data) from bidders to the auction authority, it should be used to send all security-critical information. For example, in the absence of such a communication between bidders and A, S could advertise a Vickrey auction to the bidders, but tell A that it is a first-price auction.

Avoiding replay attacks in e-voting schemes. Similar cut-and-paste replay attacks can be applied to the voting schemes that base on homomorphic cryptosystems. Here one can use exactly the same solution as in the previous paragraphs. Even if replaying is hard to mount to voting systems, our proposed defence mechanisms are so simple that one might consider using them.

7 Discussion

Local electronic auctions. In *local electronic auctions*, the bidders are physically present at an auction house, and participate via a local wireless network by using some mobile devices for computations. Local online auctions have some specific positive properties that simplify their organization and decrease the trust requirements. First, due to the locality assumption, the bidders can closely examine the goods before they decide to bid. Similarly, the winning bidder is physically present and payment can be enforced as in traditional auctions. Hence, the two most common source of complaints about Internet auctions are avoided. Second, we can assume high bandwidth capacity and sufficiently reliable communications between the seller and the bidders. In particular, the audience is captive: Bidders will stay available. Therefore, a multiple-round auction is not a problem.

Our auction schemes were designed with local electronic auctions in mind though not solely for them. Partially due to these remarks, we have assumed that law enforcement is out of the scope of the current paper: It is certainly easy to enforce correct behaviour in local auctions, but in remote auctions one must use additional protocols that are not described in this paper, e.g., punish bidders who refuse to co-operate in the confirmation phase.

Moreover, in local electronic auctions the bidder and the seller are in the same room or at least building, while the authority might be kilometers away. The same authority might be involved in many other auctions in parallel. This observation motivated us to prioritize the seller-authority communication complexity over the seller-bidders communication complexity.

Limitation on the number of valuations. A disadvantage of the homomorphic scheme from Section 5.2 is that the number of different valuations is small. Namely, if the plaintext message space is $\mathcal{M}$ then the maximum number of valuations V and the

Table 1. Example values of maximum possible B and V for some common cardinalities of message spaces. In general, a greater $|\mathcal{M}|$ means that either higher security parameter has to be used, or the bid should consist of several encryptions. "∞" means that the number of possible bidders far exceeds the population of Earth, and is hence virtually unlimited.

$\lfloor \log_2 \lvert\mathcal{M}\rvert \rfloor, V$	100	200	300	400	500	1000
1024	1209	34	10	5	4	2
1536	$4.2 \cdot 10^4$	205	34	14	8	2
2048	$1.4 \cdot 10^6$	1209	110	34	17	4
3072	$1.7 \cdot 10^9$	$4.2 \cdot 10^4$	1209	205	70	8
4096	∞	$1.4 \cdot 10^6$	12884	1209	292	17

maximum number of bidders B are bounded by $V \cdot \log_2 B < \log_2 |\mathcal{M}|$. While one can increase the size of message space, doing this will greatly increase the computational complexity of the homomorphic scheme, with $\log_2 |\mathcal{M}| = 3072$ being almost the limit with the current computational technology. Still, it means that for smaller V, the number of bidders is almost unlimited, as seen from Table 1.

We feel that choice $V \leq 500$ is sufficient in most of the auctions. For example, in an auction of a second-hand item, the bid $b \in [0, V]$ could correspond to the price $\frac{5}{V} Pb$, where P is the original price of the sold item. A price increase of 1% of P seems to be sufficiently precise. Moreover, the mapping between the bids and actual prices does not have to be linear, it only has to be strictly monotonic and publicly known. In particular, one might use higher precision with large bids than with small bids. The fact that in our scheme there might be more bidders than available bid options should not be a big concern. (Similar encoding was used in [DJ01] in the context of electronic voting where V—the number of candidates—is usually much less than 500.)

8 Comparison to Naor-Pinkas-Sumner Scheme

We will next compare the homomorphic scheme from Section 5.2 to the Naor-Pinkas-Sumner scheme [NPS99], the only previous cryptographic Vickrey auction scheme that does not rely on the threshold trust.

In our scheme, A receives more information than in [NPS99]. On the other hand, detecting misbehavior by the auction authority A is considerably more complicated in [NPS99]. Basically, to catch a cheating authority with probability $1 - 2^{-m}$, the off-line complexity in their scheme will increase m times, compared to the basic scheme. In the homomorphic scheme, the actions of A are verifiable; verifiability can be omitted but this would decrease the interaction only by a half.

First, the off-line communication complexity of Naor-Pinkas-Sumner scheme (*without* applying the cut-and-choose method) is $300B \cdot \log_2 V$ bytes, the on-line communication complexity between the servers is about $Bt \cdot \log_2 V$ and the communication complexity of each bidder is $\Theta(t \cdot \log_2 V)$. This makes the total communication overhead of the auctioneer to be $\Theta(Bt \cdot \log_2 V)$.

In the homomorphic scheme, without the confirmation phase, bidders' sole communication with S consists of one encryption and a proof of bid correctness that takes

together $(C + 5t + 4R + M)O(\log_2 V)$ bits, and requires $\Theta(\log_2 V)$ encryptions. If B is a prime then a constant number of encryptions and communication of $\Theta(V \cdot \log_2 B)$ bits suffices. Confirmation phase has the same complexity. Therefore, the total communication between the auctioneer and the bidders is $\Theta(BV \cdot \log_2 B)$.

As motivated in Section 7, our primary concern as regards communication complexity is the communication between S and A that is dominated by the noninteractive proof that $(X_2, \text{tiebreak})$ was correctly calculated. When B is a prime then the asymptotic seller-authority communication complexity will be $\Theta(V \cdot \log_2 B)$ that is close to optimal for large-scale auctions. We will next give a comparison of the seller-authority communication complexity for some concrete values of B and V. We will use the Damgård-Jurik cryptosystem, where $R \approx C \approx \frac{s+1}{s} M$. We will suppose that $M = 1024$ and choose s as $s \leftarrow \lceil V \cdot \log_2 B / M \rceil$. We will also assume that $t = 80$ and that $V \in \{300, 500\}$. For these special cases, efficiency comparison of the homomorphic scheme from Section 5.2 with the Naor-Pinkas-Sumner scheme [NPS99] is presented in Table 2. (For the homomorphic scheme, this table shows the combined total length of a range proof and a range proof in exponents, since both are used in the correctness proof. This table also presents two versions of the homomorphic scheme, one that works with a generic B and another one which works only with a prime B.)

As seen from Table 2, if $B \geq 50$ then the seller-authority interaction in the prime-B homomorphic scheme is less than $30V \cdot \log_2 B$ bytes. The prime-B homomorphic scheme and the Naor-Pinkas scheme have roughly equal communication complexity for small-scale auctions. On the other hand, the prime-B version of the homomorphic scheme is at least 10 (resp., at least 100) times more communication-efficient than the Naor-Pinkas-Sumner scheme in medium-scale auctions (resp., in large-scale auctions). The difference in interaction will be even greater when the cut-and-choose method is applied to the Naor-Pinkas-Sumner scheme.

Finally, note that in our scheme, the computations of the seller consist of a few exponentiations per every bidder and a few exponentiations when communicating with the authority. Every bidder has to do a few exponentiations. On the other hand, the computational complexity of the authority is somewhat higher due to the use of a range proof; however range proofs from [Lip01] seem to provide an adequate efficiency.

9 Conclusions

We proposed two different auction schemes that work in a setting without threshold trust and are practical for a large number of bidders. In both schemes we have an seller S and an auction authority A that are assumed not to collude. In the second, homomorphic, scheme we embed many bids in one encryption. This allows the communication complexity to be reduced substantially.

The homomorphic scheme achieves, especially compared to [NPS99], (1) Similar level of security for other parties w.r.t. seller or bidders; (2) Verifiability of A during the protocol execution: A can change the outcome of auctions only when colluding with S; (4) Both bidder-seller and seller-authority communication complexities are reduced to $\Theta(V \cdot \log_2 B)$ bits with a moderate-size hidden constant in the Θ-expression. This makes it possible to use our auction scheme in large-scale auctions.

Table 2. Communication efficiency of the homomorphic scheme (for both generic B and a prime B) and the Naor-Pinkas-Sumner scheme [NPS99] for $V \in \{300, 500\}$ and varying B. The proof lengths are given in kilobytes. In the case of our scheme we also mention the size of s.

V			3	4	8	16	32	64	128	256	512	1024
300	Our	gener. B	24.6	24.6	24.6	38.1	38.1	38.1	51.6	51.6	51.6	51.6
		prime B	5.7	5.7	5.7	8.4	8.4	8.4	11.0	11.0	11.0	11.0
			$s=1$			$s=2$			$s=3$			
	[NPS99]		7.2	9.6	19.3	38.6	77.1	154.3	308.6	617.2	1234.3	2468.6
500	Our	gener. B	26.5	26.5	41.0	41.0	55.6	55.6	70.1	70.1	84.6	84.6
		prime B	5.7	5.7	8.4	8.4	11.0	11.0	13.6	13.6	16.2	16.2
			$s=1$		$s=2$		$s=3$		$s=4$		$s=5$	
	[NPS99]		7.9	10.5	21.0	42.0	84.1	168.1	336.2	672.4	1344.9	2679.7

On the other hand, the main drawbacks of our scheme are (1) Somewhat lower level of confidentiality for other parties w.r.t. auction authority; and (2) Limited number of possible bids. However, as argued before, both drawbacks might not be that serious. In particular, we feel that scalability in the number of possible bidders is more important than in the number of possible bids. Finally, note that the homomorphic scheme from Section 5.2 can be used as a backbone for voting scheme, modulo the change that A, instead of sending back $(X_2, \text{tiebreak})$ and proving its correctness, just sends back $x = D_A(c)$ together with a proof of correct decryption.

Acknowledgments and Further Work. The first author was partially supported by Nokia Research. We would like to thank Ivan Damgård for pointing out their new range proof in exponents [DJN] and for comments on Section 4, and Benny Pinkas for useful discussions and comments. We were recently reported by the author of [Lip01] that there exists a really efficient range proof in exponents also if one does not assume that B is a prime.

From the efficiency point of view, our auction scheme is an interesting complement to the Naor-Pinkas-Sumner auction scheme, since it has communication complexity of $\Theta(V \cdot \log_2 B)$ as compared to $\Theta(B \cdot \log_2 V)$ (with a considerably smaller constant in the big-Θ expression). It would be interesting to find a secure Vickrey auction scheme without threshold trust where the communication complexity is polylogarithmic in both B and V. It would also be interesting to know how to avoid A getting to know the bid statistics without using threshold trust.

References

[BS01] Olivier Baudron and Jacques Stern. Non-interactive Private Auctions. In Paul Syverson, editor, *Financial Cryptography — Fifth International Conference*, Lecture Notes in Computer Science, Grand Cayman, BWI, 19–22 February 2001. Springer-Verlag. To appear.

[Cac99] Christian Cachin. Efficient Private Bidding and Auctions with an Oblivious Third Party. In *6th ACM Conference on Computer and Communications Security*, pages 120–127, Singapore, 1–4 November 1999. ACM Press.

[Dam01] Ivan Damgård. Personal communication, November 2001.

[DF01] Ivan Damgård and Eiichiro Fujisaki. An Integer Commitment Scheme Based on Groups with Hidden Order. Technical Report 064, IACR, 13 August 2001. Available at `http://eprint.iacr.org/2001/064/`.

[DJ01] Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography '2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, 13–15 February 2001. Springer-Verlag.

[DJN] Ivan B. Damgård, Mads J. Jurik, and Jesper B. Nielsen. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-key System. *International Journal of Information Security*. To appear. Manuscript, available from authors.

[Kik01] Hiroaki Kikuchi. $(M+1)$st-Price Auction Protocol. In Paul Syverson, editor, *Financial Cryptography — Fifth International Conference*, Lecture Notes in Computer Science, Grand Cayman, BWI, 19–22 February 2001. Springer-Verlag. To appear.

[Lip01] Helger Lipmaa. Statistical zero-knowledge proofs from diophantine equations. Cryptology ePrint Archive, Report 2001/086, 20 November 2001. `http://eprint.iacr.org/`.

[NPS99] Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy Preserving Auctions and Mechanism Design. In *The 1st ACM Conference on Electronic Commerce*, Denver, Colorado, November 1999.

[NS93] Hannu Nurmi and Arto Salomaa. Cryptographic Protocols for Vickrey Auctions. *Group Decision and Negotiation*, 2:363–373, 1993.

[Pin01] Benny Pinkas. Personal communication, October 2001.

[RH95] Michael H. Rothkopf and Ronald M. Harstad. Two Models of Bid-Taker Cheating in Vickrey Auctions. *Journal of Business*, 68(2):257–267, April 1995.

[RS86] Michael O. Rabin and Jeffrey O. Shallit. Randomized Algorithms in Number Theory. *Communications in Pure and Applied Mathematics*, 39:239–256, 1986.

[RTK90] Michael H. Rothkopf, Thomas J. Teisberg, and Edward P. Kahn. Why are Vickrey Auctions Rare? *The Journal of Political Economy*, 98(1):94–109, February 1990.

[SM00] Kouichi Sakurai and Shingo Miyazaki. An Anonymous Electronic Bidding Protocol Based on a New Convertible Group Signature Scheme. In Ed Dawson, Andrew Clark, and Colin Boyd, editors, *Fifth Australasian Conference on Information Security and Privacy*, volume 1841 of *Lecture Notes in Computer Science*, pages 385–399, Brisbane, Australia, 10–12 July 2000. Springer-Verlag. ISBN 3-540-67742-9.

[Vic61] William Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, 16(1):8–37, March 1961.

Almost Optimal Hash Sequence Traversal

Don Coppersmith[1] and Markus Jakobsson[2]

[1] IBM T.J. Watson Research Center,
Yorktown Heights, NY, 10598
[2] RSA Laboratories, Bedford, MA 01730. http://www.markus-jakobsson.com

Abstract. We introduce a novel technique for computation of consecutive preimages of hash chains. Whereas traditional techniques have a memory-times-computation complexity of $O(n)$ per output generated, the complexity of our technique is only $O(log^2 n)$, where n is the length of the chain.
Our solution is based on the same principal amortization principle as [2], and has the same asymptotic behavior as this solution. However, our solution decreases the real complexity by approximately a factor of two. Thus, the computational costs of our solution are approximately $\frac{1}{2}log_2 n$ hash function applications, using only a little more than $log_2 n$ storage cells.
A result of independent interest is the lower bounds we provide for the optimal (but to us unknown) solution to the problem we study. The bounds show that our proposed solution is very close to optimal. In particular, we show that there exists no improvement on our scheme that reduces the complexity by more than an approximate factor of two.

Keywords: amortization, hash chain, pebbles, upper and lower bounds.

1 Introduction

Hash chains have been proposed as a tool for improving the efficiency of a variety of practical and valuable cryptographic applications, e.g., [6,7,8,9,12]. However, the cost for computing the next value on a hash chain is a topic that has largely been ignored. For long chains, this task may account for a quite noticeable – if not overwhelming – portion of the total computational effort of the protocol.

The technique most oftenly employed (whether implicitly or explicitly stated) is to compute each preimage by iterative hash function application to a common seed. However, such a method – which clearly has a computational complexity of $O(n)$ for a chain of length n – is highly wasteful in that the same sequence of values is repetitively computed. Another possible method, in which all values are precomputed and stored, substantially reduces the on-line computational cost, but has a staggering storage complexity of $O(n)$. All straightforward combinations of these two techniques can be shown to have a memory-times-computation complexity of $O(n)$, which is often beyond the reasonable for desirable parameter choices. As an example, one can see that such a high complexity would be punitive in protocols like the broadcast authentication protocols by Perrig *et al.*

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 102–119, 2003.

[7,8,9]. There, the delay between the transmission of a packet and the receiver's authenticity check of the same is determined by the amount of time between the release of consecutive hash preimages. This makes short time periods (i.e., rapid hash chain traversal) desirable. At the same time, since their method is intended for very inexpensive devices, the permissible "allowances" for computation and storage are strict. Together, these restrictions drastically limit the possible lifetime of such broadcast devices.

Influenced by amortization techniques proposed by Itkis and Reyzin [1], Jakobsson [2,3] showed how to reduce the above mentioned memory-times-storage complexity to $O(log^2 n)$. This was accomplished by the introduction of a technique in which multiple intermediary values are kept, but their values (and positions in the chain) constantly modified. While the protocol of Jakobsson stressed simplicity over efficiency, we take the opposite approach in this paper. By introducing a set of tricks and new techniques, we are able to improve the efficiency at the expense of simplicity – the latter both in terms of the protocol and its associated proofs. Thus, our improved protocol allows for a reduction of the computational requirements to slightly less than half of [2], while slightly reducing the storage demands for most practical parameter choices.

Using a numeric example to illustrate the differences, we have that consecutive preimages of a chain of length 2^{31} can be generated using 31 hash function applications and 868 bytes of storage in [2], while our algorithm only needs 15 hash function applications and 720 bytes of storage – both assuming the use of SHA [11]. More generally, the computational requirements of our protocol are $\lfloor log_2 \sqrt{n} \rfloor$ hash function evaluations per output element. The storage requirements, in turn, are $\lceil log_2 n \rceil + \lceil log_2(log_2 n + 1) \rceil$ memory cells, where each cell stores one hash chain value and some short state information. Note that these are *not* average costs, but upper bounds on the cost per element that is output, given a hash chain of length n.

In order to allow for these savings, it is necessary to shift from a strategy with a very simple movement pattern to a more complicated strategy where the per-element budget always is exhausted. (Another way to think about it is that the reduction of the budget demands a wiser spending strategy under which no resources are wasted.) Consequently, we develop a rationale for how to assign a now reduced budget among a set of pebbles whose demands and priorities are modified over time. Furthermore, we propose a protocol based on these design criteria and show it to be correct.

Finally, we show that our strategy, and the related protocol we propose, are near optimal. We do this by providing upper and lower bounds for the most efficient algorithm possible, and compare the resulting complexity to that of our protocol.

2 Intuition

The aim of our protocol is to compute and output a series of consecutive values on a hash chain with minimal memory and computational requirements. We call the

two ends of our chain "the beginning" and "the end" of the chain, functionally corresponding to the public vs. the secret keys of the scheme. In a setup phase, the chain is constructed by choosing the end value at random and iteratively applying the one-way function to get the value at the beginning of the chain. We want to output all the values of the chain – starting with the value at the beginning of the chain, and ending (not surprisingly perhaps) with the value at the end of the chain. Each value in the chain (except the end value) is the hash one-way function of the adjacent value in the direction of the end of the chain. In other words, each output value is the hash preimage of the previously output value. Therefore, previously output values are not useful in computing the next value to be output, which instead has to be computed by iterative application of the hash one-way function to a value towards the end of the chain.

Our solution employs novel amortization techniques to reduce the worst case computational cost per element to the average cost. Simply put, we use the known principle of conserving resources by not letting anything go to waste. Technically speaking, this is done by assigning a computational budget per step in the chain, and applying any "leftover computation" towards the computation of future elements. The contribution of this paper is a technique for computing the desired sequence without ever exceeding the per-step budget, along with the establishment of the required computational budget and memory demands.

In order to reach our goals, the leftover computation we apply towards future elements to be computed must be sufficient to compute these. More importantly, it must be sufficient to compute them *on time*. Thus, at each point in the chain, the cumulative required expenditures must not exceed the cumulative computational budget.

Assume that we want to compute a value close to the beginning of the chain. If no values along the chain are stored, then we have to perform an amount of work proportional to the length of the chain, which we denote n. Let us now introduce one "helper value" at some distance d from the current chain element. Then, the cost of computing the current value is that of $d-1$ hash function evaluations. The cost for the next value to be computed, in turn, will be $d-2$ such evaluations. However, once the helper value is reached, the cost of the next value will be that of reaching the endpoint of the chain – assuming we only employ one helper value. One can see that the total cost is minimized if $d = n/2$, i.e., the helper value is located on the mid-point of the chain.

If we can use *two* helper points (which corresponds to storing two extra elements instead of one) then one could let the entire interval be split into three equally long intervals, in which case the cost of computing the next element would be upper bounded by $n/3$ hash function evaluations. On the other hand, if we first split the entire interval in two equally long intervals, and then split the first of these two into two sub-intervals, then we have upper bounded the *initial* computational cost at $n/4$ hash function evaluations. This lower cost applies to the first half of the entire interval, after which the distance to the next element (which is the endpoint) would be $n/2$. However, if we – once we reach the first helper point – relocate this to the midpoint between the "global midpoint" and

the endpoint, we will maintain an upper bound of $n/4$. (See figure 1 for an illustration of how the relocation occurs.)

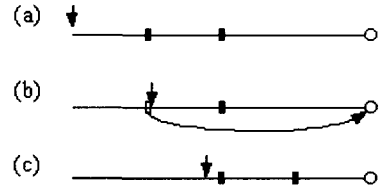


Fig. 1. The figure shows the movement of helper values (squares) as the current position (small arrow) changes. In (a) the positions right after setup are shown; (b) shows the relocation of the first helper value as it has been reached. In (c), its relocation is completed.

This assumes that we have enough remaining computation for this relocation. Note now that if we have *three* helper points, we have more values to relocate, but we also reduce the computational upper bound for each element (since the intervals decrease in length with an increasing number.) We can see that using approximately $\log n$ helper values, we will maximize the benefits of the helper points, as we then, for every element to be computed, will have a helper point at a maximum distance of two steps away.

If we use this approximate number of helper values, the cost of computing the next value to be output is (on average) that of half a hash function evaluation, making the budget for relocation the dominating portion of the required total budget. When we relocate a helper point, the cost for computing its value at the wanted location is proportional to the distance (in the direction of the endpoint) to the next known value, whether this is a helper value or the endpoint itself. It is clear that the more helper points we employ, the lower this cost will be. However, the cost metric we are interested in minimizing is not computation alone, but the product of computational complexity and memory complexity.

For each element to be computed and output we assign a budget, corresponding to the computational upper bound per element. The computation of the next element has the highest priority to access this budget, since for each step, one has to compute and output the appropriate element. Any leftovers are assigned to the computation of helper values. These are partitioned into *high priority* helper values and *low priority* helper values. High priority helper values are relocated into already rather small intervals, located close to the current element (i.e., the element to be output in the current round). Low priority helper values, in turn, traverse larger distances, and further from the current element. The low priority helper values are only assigned those portions of the budget that remain after the current element has been computed, and the high priority helper values have exhausted their needs (i.e., arrived at their respective destinations.) Given

that low priority helper values are traversing distances large enough to make it impossible for them to both get started and arrive in one and the same time interval, we can with only two such helper values guarantee that there always will be one left to "soak up" any computational leftovers.

During the setup phase, the endpoint of the chain is randomly selected, and the start point obtained by iterated hash function evaluation. This may be done by a device with less computational limitations than the device that later will compute and output the consecutive elements. During setup, the helper values will also be initialized. The first helper value will be placed at the mid-point between the endpoint and the starting point, thereby splitting the entire interval in two. The ith helper value will be placed on the midpoint between the position of the $i-1$st helper value and the starting point. Thus, each helper value will have a position and a value associated with them, where the value is obtained by iterated hash function application of the endpoint value or a previously placed helper value.

We show that our protocol is *almost optimal.* We do this by providing upper and lower bounds for the optimal solution to the problem. We wish to point out that while we do not know what the optimal solution is, we know that our solution (which is not the optimal) is "as good as one can get" – for all practical purposes. This does not count potential improvements leading to a simpler or shorter algorithm, but only refers to its memory and computational complexity.

Outline: We begin by introducing our method for computing the sequence of hash preimages, first laying out the preliminaries (section 3) and then elaborating on the protocol (section 4). In section 5, we present and prove our claims relating to the completeness and correctness of the protocol, and relating to the upper and lower bounds of the optimal solution to the problem we study.

3 Preliminaries

Definitions

We use the term *hash chain* H to mean a sequence of values $< v_0, v_1, \ldots, v_i, \ldots v_n >$, where v_n is a value chosen uniformly at random from $\{0,1\}^l$, and $v_i = h(v_{i+1})$, where $h : \{0,1\}^* \rightarrow \{0,1\}^l$ is a hash function or another publicly computable one-way function. We refer to v_0 as the *starting point*, and to v_n as the *endpoint.*

We define the *span* n to be the length of the hash chain, i.e., the number of elements in the sequence to be generated. We assume that $n = 2^\sigma$ for some integer $\sigma > 2$.

We define the *budget* b as the number of computational units allowed per element of the sequence that is output. Here, we only count hash function evaluations and not other computational steps associated with the protocol execution. This is reasonable given the fact that the computational effort of performing one hash function evaluation far exceeds the remaining work per step.

We refer to each helper value, and to the endpoint, as a *pebble*. Each pebble p_j has a position in the chain and a value associated with itself. The position of the start value is zero, and the position of the endpoint equals the span n. If *position* is the position of a pebble, then its value is $v_{position}$. Additionally, each pebble is associated with a *destination* (the position to which it is going); a *priority* (high or low); and an activity *status* (free, ready, active, arrived.) Here, pebbles that are not in use are referred to as *free*; these may be allocated to a particular task once the need arises. A pebble that is *ready* has been assigned a task, but has not yet started to move, while an *active* pebble in in motion. We use the ready state for a so-called "backup" pebble. This is a pebble that will become a low-priority pebble as soon as the active low-priority pebble reaches its destination. Finally, a pebble is assigned status *arrived* if it located at its destination point, and is still needed there (i.e., has not yet been reached by the "current" pointer, which corresponds to the position of the current output value.) We let k denote the number of pebbles used; the amount of storage needed is k times the amount needed per value, plus the amount (registers, etc.) needed for the execution of the protocol.

Goal

After performing a setup phase, we wish to generate the sequence H, element by element (and starting from v_1), using a minimal budget and a minimal number of pebbles. We will demonstrate a method with required budget $b = \lfloor \sigma/2 \rfloor$ and using $k = \sigma + \lceil log_2(\sigma+1) \rceil$ pebbles, where $n = 2^\sigma$ is the number of elements of H.

Design Guidelines

If a pebble is located at the position corresponding to the current output, we say that this pebble has been *reached*, at which time it receives "free" status. All pebbles with status free are assigned a new position, destination, state and priority, according to guidelines that are set up to guarantee protocol completeness. To explain the proposed protocol, we present these guidelines along with their technical motivations.

"First things first". At each step, we first compute and output the appropriate hash chain value (which we call the *current* value); then, any remaining budget is assigned to active high-priority pebbles, starting with the pebble with the lowest position value (i.e., closest to the position associated with the current output value.) First then, any still remaining budget is assigned to active low-priority pebbles. This is to ensure that computational results that soon will be needed are ready on time.

Controlling high-priority pebbles. The high-priority pebbles are started at the "far end" of the first interval after the current value that does not already contain an active pebble, counting only intervals of size greater than two. In other words, if there is a pebble in *free* state, this will obtain the position and

value of the first interval of size four or greater in which there is no active pebble, and will be given state *active*. Here, pebbles that just have been assigned to a position are considered to be in the interval in question. When the pebble reaches its destination (at the mid-point of the interval), it is given state *arrived*. Thus, if the resulting intervals (half the size of the original interval) are at least of size four, a new pebble may immediately be assigned a starting position equalling the position of the pebble that just reached its destination.

High-priority pebbles are only allowed to be active in positions lower than the active low-priority pebble, and are otherwise kept in *free* state. This is to make sure that high-priority pebbles do not take too much of the available budget: it slows them down to the benefit of low-priority pebbles when they complete their imminent tasks.

Controlling low-priority pebbles. We want there always to be a low-priority pebble that can "soak up" any remaining computational budget. We can achieve this by (1) having one "backup" pebble that is not assigned to any task, but which is ready to become the active low-priority pebble; and by (2) making each low-priority pebble traverse a distance that is sufficiently long that it and its backup cannot both complete before a new pebble becomes available. (When a new pebble does become available, it will be assigned to become a backup pebble if there is none, otherwise a high-priority pebble.)

According to our protocol, pebbles close to the current pointer have destinations set two steps apart. Therefore, assuming they will arrive in a timely fashion, they will be "virtually spaced" two steps from each other. Thus, a pebble will be reached by the current pointer every two moves (where a move is the computation performed between two consecutive outputs). If the distance low-priority pebbles need to travel from their inception until they reach their destination is at least twice the budget per move, then a new pebble will always be reached and relocated before the low-distance pebble and its backup reach their goals. Therefore, if the backup low-priority pebble is converted to an active low-priority pebble, a new backup pebble will be created before the converted pebble reaches its goal. Thus, our requirement will be satisfied.

By taking this approach, we can guarantee that the entire budget of each step will always be consumed, since there will always be an active low-priority pebble. According to our suggested approach, we only need *one* active low-priority pebble at the time, and one "backup" low-priority pebble.

4 Protocol

Setup. The endpoint v_n is chosen uniformly at random from $\{0,1\}^l$, where we may set $l = 160$. The sequence $H =< v_0, v_1, \ldots, v_i, \ldots, v_n >$ is computed by iterated application of the hash function h, where $v_i = h(v_{i+1})$, $0 \leq i < n-1$.

Pebble p_j, $1 \leq j \leq \sigma$, for $\sigma = log_2\, n$, is initialized as follows:

$$\begin{cases} position \leftarrow 2^j \\ destination \leftarrow 2^j \\ value \leftarrow v_{2^j} \\ status \leftarrow \text{arrived.} \end{cases}$$

The remaining pebbles, p_j, $\sigma < j \leq k$, have their status set to free. All the pebble information is stored on the device we wish to later generate the hash sequence; this device also stores counters $current \leftarrow 0$ and $backup \leftarrow 0$, along with the span n. The pair $(startpoint, current) = (v_0, 0)$ is output. The starting point v_0 corresponds functionally to the public key of the chain.

Maintenance. In the following, we assume that the pebbles p_j, $1 \leq j \leq k$, are kept sorted with respect to their destination, with the lowest destination value first; and that pebbles that do not have a destination assigned appear last in the ordered list. Consequently, the next pebble to be reached (from the current position) will always be p_1. When the status of the pebbles is changed at any point, the altered item is inserted at the appropriate place in this sorted list. We let LP (short for low priority) be an alias of the active low-priority pebble. Thus, $LP.position$ is the current position of the active low-priority pebble, independently of what pebble number this corresponds to. Similarly, BU refers to the backup low-priority pebble.

Generation. The following protocol is performed in order to generate the hash sequence; each iteration of the protocol causes the next hash sequence value to be generated and output. The protocol makes use of two routines, $placeHP$ and $placeLP$; these assign values to high priority resp. low priority pebbles according to the previously given intuition. The corresponding algorithms will be described after the main routine is presented:

1. Set $available \leftarrow b$. *(Set the remaining budget.)*
2. Increase $current$ by 1.
3. If $current$ is odd then *(No pebble at this position.)*
   output $h(p_1.value)$, *(Compute and output.)*
   decrease $available$ by 1,
   else *(A pebble at this position.)*
   output $p_1.value$, *(Output value, set pebble free.)*
   set $p_1.status \leftarrow$ free,
   if $current = n$, then halt. *(Last value in sequence.)*
4. For all free pebbles p_j do *(Reassign free pebbles.)*
   if $backup = 0$ then *(Backup low-priority needed.)*
   $p_j.priority \leftarrow$ low,
   $p_j.status \leftarrow$ ready,
   $BU \leftarrow p_j$,
   $backup \leftarrow 1$,
   else
   Call `placeHP` (p_j) *(Make it high priority.)*
5. Sort pebbles.
6. Set $j \leftarrow 1$. *(First pebble first.)*

```
7. While available > 0 do
    if p_j.status = active then                 (Only move active pebbles.)
        decrease p_j.position by 1,             (Update its position...)
        p_j.value ← h(p_j.value),               (... and value...)
        decrease available by 1,                (... and do the accounting.)
        if p_j.position = p_j.destination then  (Pebble arrived!)
            p_j.status ← arrived,
            if p_j.priority = low then          (A low-priority pebble arrived.)
                LP ← BU,                        (Backup becomes low priority.)
                backup ← 0,
                Call placeLP,                   (Activate new low priority pebble!)
                Sort pebbles.
    increase j by 1.                            (Next pebble!)
8. Sort pebbles.
9. Go to 1.                                     (Next element now.)
```

Routine PlaceLP. We begin by describing how one could compute the sequence of values assigned to variables during calls to PlaceLP. (We later describe how just *one* such assignment can be computed, as opposed to an entire sequence. We also elaborate on a method that is less wasteful of stack space.) The wanted functionality of the routine is to compute the next starting point for a low-priority pebble, along with the associated destination.

In the following, we use "normalized" positions for ease of reading and uniformity over different spans. To get a real position from a normalized position, one multiples the latter by a value λ, where λ is the smallest power of two not smaller than $2b$, and where b is the budget. In other words, $\lambda = 2^{\lceil log_2 b \rceil + 1}$. Thus, the series of normalized starting points, starting with $(4, 8, 6, 8, 16, 12, 10, 12, 16, 14, 16)$, corresponds to a series $(32, 64, 48, 64, 128, 96, 80, 96, 128, 112, 128)$ for $b = 4$, $\lambda = 8$. Similarly, the destination points and the distances between the starting points for the pebbles and their destinations are described in normalized terms.

When a free pebble is activized, it is placed on top of another pebble (located at its starting point) and given a destination. The destination is at the mid-point of the interval to the next pebble (in the direction of the current pointer). Thus, the intervals are split in two; as the pebble arrives, a new pebble is placed on top of it, with a destination of the next lower midpoint. If an interval is too small to be split, the pebble is instead placed at the end of the *next* interval.

We associate the first split of an interval with the root of a tree. The children of the root correspond to the splits of the resulting two intervals, and *their* children by their respective splits. The leaves correspond to the smallest splits of the intervals. Figure 2 shows the nodes of a small tree, and their order of traversal, and how these corresponds to the order of pebble placement in the hash chain. Figure 4 shows the starting points and destinations for one example tree, according to the assignment strategy described below.

We say that the height of a tree is the number of layers of nodes it has. (Thus, a tree consisting on only one node has height 1.) With each node of the tree, we associate a starting point; a distance; and a destination, where the destination is the difference between the starting point and the distance.

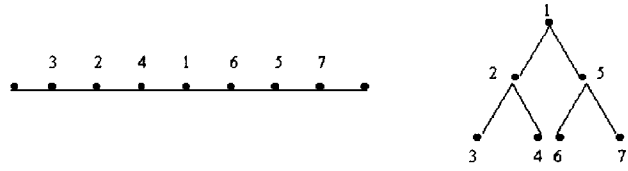


Fig. 2. The left part of the figure shows the desired destinations of pebbles, where the number corresponds to the relative order of assignment. The relative order therefore corresponds to the depth-first traversal order of the corresponding tree, shown to the right. Note that the hash chain nodes correspond to a vertical projection of the tree nodes.

The normalized *starting point* for the root of a tree of height j is $start = 2^{j+1}$. The normalized *distance* of a node at height i in the tree is $dist = 2^{i-1}$; thus the distance of the root is 2^{j-1}, and leaves of the tree all have normalized distance $dist = 1$. The normalized *destination* $dest$ of any node (and the root in particular) is the difference between its starting point and distance, i.e., $dest = start - dist$. Finally, the starting point for a left child is its parent's destination value, $parent.dest$, while it is the parent's starting value $parent.start$ for a right child.

Consider the sequence of assignments of *start* and *dest* that one obtains from performing a depth first search of a given tree (with the left child always traversed before the right child). That is the sequence of assignments corresponding to the associated initial interval (i.e., the interval before splitting), as illustrated in figure 4. Consider further the sequence of such assignments one gets from traversing a forest of such trees, where the first tree has height one, and each tree is one level higher than its predecessor. That is the sequence of normalized assignments we need in our protocol.

Each call to PlaceLP first computes such a pair of normalized values, all from the above described sequence; these are then multiplied by λ and the product returned as the result of the function. Thus, it sets

$$\begin{cases} LP.priority \leftarrow low \\ LP.status \leftarrow active \\ LP.position \leftarrow \lambda\, start \\ LP.destination \leftarrow \lambda\, dest \end{cases}$$

As soon as $\lambda\, start > n$, no assignment is performed, since there is no need for low priority pebbles any longer. Any calls to `PlaceLP` after that return without any assignment.

Claim: The routine `PlaceLP` generates a sequence of elements, where the ith element of the sequence corresponds to the pair of starting position and destination of the ith low-priority pebble to be activated. The starting point corresponds to that of a pebble already placed on the hash chain, and the destination corresponds to the middle point between this same pebble and the closest pebble

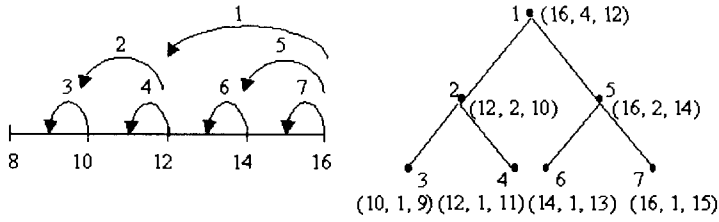


Fig. 3. The left part of the figure shows the desired movement pattern of low priority pebbles in the interval between normalized positions 8 and 16, where the numbers on the arrows correspond to the relative order of the movements. The right part of the figure shows the tree structure from which these movements are obtained. The node number corresponds to the relative order of the movement, and is obtained by depth-first traversal of the tree. The triple associated with a node is the (starting point, distance, destination) of the corresponding pebble. A tree of height 2 is used for the normalized interval 4-8, and one of height 1 for the interval 2-4. Similarly, larger trees are used for later intervals.

in the direction of the current pointer. The distance between starting point and destination is at least twice the budget per step that is available for moving pebbles, which guarantees that a recently placed low priority pebble does not reach its destination before another pebble has been activated. Once such a pebble does reach its destination, a "backup low-priority pebble" is activated by being turned into a standard (active) low-priority pebble. Once a new pebble is reached by the current pointer (at which time it is redundant in its current position, and needs to be relocated), such a pebble is made into a backup low-priority pebble, if the old such pebble has been activated.

Routine PlaceHP. The routine for computing the next location for a high-priority pebble is similar to the above, its main differences being that (1) the real and normalized values coincide, and (2) after the trees in the forest have reached height $log_2 \lambda$, they stop growing. Here, as before, $\lambda = 2^{\lceil log_2 b \rceil + 1}$, where b is the budget per output element produced.

The starting position of the jth tree is $start = 2^{j+1}$ for $j \leq log_2 \lambda$, and $start = \lambda(j - log_2 \lambda)$ for $j > log_2 \lambda$. As before, the starting point for a left child is its parent's destination value, *parent.dest*, while it is the parent's starting value *parent.start* for a right child. The *distance* of a node at height i in the tree is $dist = 2^{i-1}$. The *destination*, as before, is the difference between its starting point and destination, i.e., $dest = start - dist$.

Before any assignment is made, it is verified that $start \leq LP.position$, i.e., that the assignment in question is to a position before the active low priority pebble. If this is not the case, no assignment is made at this point, and the comparison re-performed at the next call to the routine. Otherwise, the following assignment is made to parameter p_j to the routine:

$$\begin{cases} p_j.priority \leftarrow \text{high} \\ p_j.status \leftarrow \text{active} \\ p_j.position \leftarrow start \\ p_j.destination \leftarrow dest \end{cases}$$

Claim: The routine `PlaceHP` generates a sequence of elements, where the ith element of the sequence corresponds to the pair of starting position and destination of the ith high-priority pebble to be activated. The starting point corresponds to that of a pebble already placed on the hash chain, and the destination corresponds to the middle point between this same pebble and the closest pebble in the direction of the current pointer. The starting point is chosen as a point between the current pointer and the active low-priority pebble, as close to the current pointer as possible, such that the distance between the starting point and the destination is at least two.

Memory Complexity. In order to conserve working space, one can opt for a solution that does not use a stack, but which recomputes the state from scratch when needed. One variable would store the height of the tree; one would store the number of steps (using depth-first search) from the root. Additionally, we need variables for *start*, *dist* and *dest*. To compute these values, we would (at each time) begin with their starting assignments, and then modify them according to the tree traversals leading to the wanted number of steps from the root. This is done in accordance with the respective assignments, as above.

The maximum tree height for `PlaceLP` is $\sigma - log_2 \lambda - 1$, since this corresponds to a normalized starting point of $2^{\sigma-\lambda}$ and a starting point of 2^σ. Thus, this variable needs $\lfloor log_2 \sigma \rfloor$ bits. For `PlaceHP`, the maximum height is $log_2 \lambda$, requiring $log_2 \lceil log_2 \lceil \lambda \rceil \rceil$ bits of storage. A tree of the maximum height has $2^{\sigma - log_2 \lambda - 1} - 1$ nodes for `PlaceLP`, and $2^\lambda - 1$ nodes for `PlaceHP`. Thus, the distances from the root can be represented with $\sigma - log_2 \lambda - 1$ respective λ bits. Finally, the maximum value of the last three variables is σ bits for each one of them, since the maximum value they can be assigned is 2^σ. (These only keep state within a computation, and so, we do not need one set for each algorithm.)

Therefore, the memory requirements of `PlaceLP` and `PlaceHP` are less than $4\sigma + log_2 \sigma + \lambda - 1$ bits. This is dwarfed by the memory requirements for the pebbles, requiring 160 bits each, resulting in a total of $160(\sigma + \lceil log_2(\sigma + 1) \rceil)$ bits.

5 Claims

Consider a span $n = 2^\sigma$, a budget $b = \lfloor \sigma/2 \rfloor$ and k pebbles, for $k = \sigma + \lceil log_2(\sigma + 1) \rceil$.

We refer to the sum of the budgets from the setup stage until a particular step as the *cumulative budget* at the step in question. We say that the protocol with a budget restriction of b and a storage restriction of k *succeeds* at step j if and only if it outputs the jth value v_j of the hash sequence during this step,

and that the protocol succeeded at step $j-1$. The protocol is said to succeed (by definition, and due to the setup procedure) at step 0 – this corresponds to the setup phase, on which we do not place any strict restrictions in terms of computation and storage.

Theorem 1: *(Completeness.)* The protocol succeeds at step j, $1 \leq j \leq n$, for a span $n = 2^\sigma$, budget $b = \lfloor \sigma/2 \rfloor$ and $k = \sigma + \lceil log_2(\sigma + 1) \rceil$ pebbles.

The proof of the theorem will be based on the following lemmata, all of which relate to the above assignments for n, b and k. Recall that λ is defined to be the smallest power of two not smaller than $2b$.

Lemma 1: *(Bootstrapping.)* The protocol succeeds at step j, $1 \leq j \leq \lambda$.

Proof of Lemma 1:
We will consider two cases: $\lambda < 8$, and $\lambda \geq 8$. Recall that the cost of moving one step ahead equals the distance from the position we move to the next (forward) pebble. Recall also that λ is defined to be the smallest power of two equal to or larger than $2b$, and thus, $\lambda < 4b$.

The first case corresponds to the two possibilities $\lambda = 2$ or $\lambda = 4$. For $\lambda = 2$, we have $b = 1$. We need budget 1 to compute v_1 from v_2 (where there is a pebble), and zero budget to obtain v_2. For $\lambda = 4$, we know that $b = 2$. Since there are pebbles at positions 2 and 4, the budgets to reach these are zero, and the budget to reach step 1 and 3 from the step before is one. Therefore, the lemma holds for the first case.

For the second case, i.e., $\lambda \geq 8$, we know that there is one pebble at λ, one at $\lambda/2$, and one at $\lambda/4$, since we start with pebbles at each position that is a power of two, and which is in the interval between 0 and n. Consider the first two quarters of the interval between 0 and λ. Notice first that the budget will be sufficient for each step of these two quarters, since the pebbles are spaced $\lambda/4$ apart, and the budget is $b > \lambda/4$ (given how λ is defined). The traversal cost for both of the quarters is upper bounded by $(\lambda/4-1)+(\lambda/4-2)+\ldots+2+1+0 = (\lambda/4-1)\lambda/8$, giving a total of $(\lambda/4-1)\lambda/4$. The total budget assignment for this first half is $(\lambda/2)b$. Since $b > \lambda/4$, we have that the cumulative budget is $(\lambda/2)\lambda/4$. Thus, the budget "surplus" for the first half is at least $(\lambda/2)\lambda/4 - (\lambda/4-1)\lambda/4 = (\lambda/4+1)\lambda/4 > \lambda/4$.

This surplus will be applied to moving pebbles *outside* the first half of the interval, since any pebble movement *inside* the first half could only lower the expenditures for this portion, which would cause an even larger surplus. The surplus, in turn, is always spent on moving pebbles, with the closest high-priority pebbles receiving priority. The first pebble outside the first half of the interval would start at λ, and have destination $3\lambda/4$. Given that the surplus of the first half of the interval is at least $\lambda/4$, the pebble would reach its destination by the time the current position is $\lambda/2$.

Consider now the third and fourth quarter of the interval between 0 and λ. We have concluded that when the current position is $\lambda/2$, there are pebbles at both position $3\lambda/4$ and position λ (where the latter pebble has been there since the setup phase.) We know that $b > \lambda/4$. Thus, the budget will be sufficient for

each step of the two last quarters of the interval, and we see that the protocol will succeed at step λ. This concludes the proof.

Lemma 2: *(Discrete points.)* The protocol succeeds at step $j = 2^i\lambda + 1$, $0 \leq i \leq \sigma - log_2\,\lambda$, if it succeeds at step $2^i\lambda$.

Proof of Lemma 2:
Assume that we have arrived at position $j = 2^i\lambda$ for some $0 \leq i \leq \sigma - log_2\,\lambda - 1$. (This means that the protocol succeeded up until this point.) We wish to prove that the protocol will succeed for the next step, too. In order for this to occur, the next pebble must be at most $b+1$ steps away from step j, or the budget will not suffice. We will show that there will be a pebble at $j+2$, which would make the lemma hold, since $b \geq 1$.

At the time when we are at step j, the total incurred costs for pebbles equal the cost for filling the space between 0 and j with pebbles, plus that for populating (with exponentially increasing intervals) the interval between j and $2j$. We will argue that this sum is equal to the cost of populating the interval between 0 and j only, *were this interval empty.* This will result in a simpler way of determining the total required pebble expenditure up until step j.

Consider the location of all pebbles in the interval between 0 and j *at the start time.* These pebbles are located at positions $2, 4, 8, \ldots, j/2$. We want pebbles at positions $j+2, j+4, j+8, \ldots, j+j/2$ when we are at position j. Due to the setup, there will already be a pebble at $2j = 2^{i+1}\lambda$. The cost of placing the pebbles at $(j+2, j+4, j+8, \ldots, j+j/2)$, given the pebble at $2j$, is identical to the cost we would have incurred if we wanted to place pebbles at $2, 4, 8, \ldots, j/2$, given the pebble at j, since the relative distances from j resp. from $2j$ are identical for the two sequences. We can therefore substitute the cost for the real sequence between j and $2j$ by the hypothetical cost for the interval 0 to j.

Therefore, the total pebble expenditures at step j will equal the total pebble expenditures we were to incur if we were to fill an empty interval between 0 and j. The cost of filling an empty interval between 0 and j involves moving one pebble a distance $j/2$; two pebbles a distance $j/4$ each, four pebbles a distance of $j/8$ each, etc. Filling the space means that (over time) every even position in the interval has a pebble. Thus, the total pebble expenditure is $\sum_{\kappa=1}^{log_2 j - 1} 2^{\kappa-1} j 2^{-\kappa}$, where $2^{\kappa-1}$ is the cardinality and $j2^{-\kappa}$ is the associated cost. This total expenditure can be seen to equal $j/2(log_2\, j - 1)$.

If at any time we are either one or two steps from a pebble (where our position corresponds to the value being output), then the total expenditures for moving a step ahead is either zero (if we are at a position with an odd number, meaning next position has a pebble) or one (if we are at an evenly numbered position.) Thus, the total "stepping" expenditures up until step j are $j/2$.

We see that the total expenditures up until step $j = 2^i\lambda$ would be $j/2(log_2\, j - 1) + j/2$ in order for there to be pebbles at positions $(j+2, j+4, \ldots, j+j/2)$ at the time the current position reaches step j. This equals $\frac{1}{2} j log_2\, j$. Given that for each of the j steps, we are assigned a budget b, the total budget up until that point will be jb. We will be successful if $b \geq \frac{1}{2} log_2\, j$. This quantity will be the largest for the end of the interval that the proof is valid for, i.e., $i = \sigma - log_2\,\lambda - 1$.

Plugging in i in the formula for j gives us $j = 2^{\sigma - log_2 \lambda - 1}\lambda = 2^{\sigma-1}$. Thus, if $b \geq \frac{1}{2}(\sigma - 1)$, then the lemma holds. According to the specifications, we have $b = \lfloor \sigma/2 \rfloor$. Therefore, if the protocol succeeds at $j = 2^i\lambda$, then it succeeds at $j + 1 = 2^i\lambda + 1$, for $0 \leq j \leq n/2$, which concludes the proof.

Lemma 3: *(Intervals.)* The protocol succeeds at step $j = 2^{i+1}\lambda$, if it succeeds at step $2^i\lambda + 1$.

Proof of Lemma 3:
Assume that the protocol succeeds at step $j = 2^i\lambda + 1$. We wish to prove that then, the protocol also succeeds at step $2j - 2 = 2^{i+1}\lambda$. This will be shown using a symmetry argument. Consider intervals of size $2^i\lambda$. Consider first such an interval starting at position 1 and ending at $2^i\lambda$, and then one starting at position $2^i\lambda + 1$ and ending at $2^{i+1}\lambda$.

Assume that the current position is at the beginning of the second interval. Assume further that the relative positions of the pebbles in the second interval (in relation to the current position) are identical to the relative positions of the pebbles in the *first* interval (in relation to the current pointer when located in the beginning of the first interval). Then, the required expenditures to reach the end of the second interval from its beginning must equal the required expenditures to reach the end of the first. This is so since the resource allocation strategy is the same for both intervals, namely that pebbles close to the current position are given priority over pebbles further away. Therefore, if the budget is sufficient to reach the end-point of the first interval, then it is also sufficient to reach the end-point of the second.

We know that the first interval will have pebbles at positions $2, 4, 8, \ldots, j$ when we are at position 1. As was shown in Lemma 2, the cumulative budget available to pebbles at step j is sufficient for the placement of pebbles at positions $j + 2, j + 4, j + 8, \ldots j + j/2$, and we know that there already is one (due to the setup) at position $2j$. Therefore, the required expenditures in the interval between $2^i\lambda + 1$ and $2^{i+1}\lambda$ are the same as those for the interval between 1 and $2^i\lambda$. This concludes the proof.

Proof of Theorem 1: The theorem follows from the above lemmata. Lemma 1 establishes that the protocol succeeds for j, $1 \leq j \leq \lambda$. Then, Lemma 2 shows that it succeeds for $\lambda + 1$ (i.e., setting $i = 0$), and lemma 3 shows that it succeeds for all values up until 2λ (again using $i = 0$). Then, using $i = 1$, lemmata 2 and 3 establish that the protocol succeeds up to position 4λ. We apply lemmata 2 and 3 iteratively, and for increasing values of i, ending with $i = \sigma - 1 - log_2 \lambda$, finally establishing that the protocol succeeds for $j = n$, which completes the proof.

In the following, we show that our solution is *almost optimal.* More particularly, we consider the product of the number of hash function evaluations needed, and the number of storage cells required, where each storage cell holds one hash chain value and some short state information. Then, the complexity of the optimal solution is $\frac{1}{4}log^2 n$ per output element, while the complexity of our solution is *approximately* $\frac{1}{2}log^2 n$ – more precisely, it is $\frac{1}{2}\lfloor log\, n \rfloor (\lceil log_2 n \rceil + \lceil log_2(log_2 n + 1) \rceil)$. Thus, we are (practically speaking) no more than a factor

of two away from the optimal solution in terms of computation-times-storage complexity.

Theorem 2: *(Lower bound.)* The optimal solution to the problem has a memory-times-computational complexity of at least $\frac{1}{4k}\lg^2 n$, where n is the length of the hash chain and k is the number of pebbles.

Proof of Theorem 2: We wish to show that the cumulative budget – the total cost of processing a string of length n using k pebbles – is at least $\frac{n}{4k}\lg^2 n$, which implies that the amortized cost per evaluation is at least $\frac{1}{4k}\lg^2 n$. This, in turn, will imply that the budget (worst-case cost per evaluation) is also at least $\frac{1}{4k}\lg^2 n$. The optimal case is $k = \frac{1}{2}\lg n$, where the amortized cost per evaluation is also $\frac{1}{2}\lg n$.

Let $g(n,k)$ be the required cumulative budget for covering a string of length n with k pebbles, excluding the initial cost (n) of setting up the pebbles.

Suppose that the furthest pebble is at position $n-T$, at the last time that it is used (cloned or used directly). Then we must cover an interval of length $n-T$ with $k-1$ pebbles (not using the kth pebble, which is stuck at position n), expend energy T to lay down the pebbles in the remaining interval, and cover that last interval with k pebbles. Optimizing over choice of T, we would have

$$g(n,k) = \min_T [g(n-T,k-1) + T + g(T,k)].$$

We want to show, by induction, that $g(n,k) \geq \frac{n\lg^2 n}{4k}$.

The inductive step will go through if we can show that, for all n, T, k, we have $h(k,T) \geq 0$, where for fixed n we define

$$h(k,T) = -\frac{n\lg^2 n}{4k} + \frac{(n-T)\lg^2(n-T)}{4(k-1)} + T + \frac{T\lg^2 T}{4k}.$$

For convenience we set $L = \lg n$. We evaluate h and its derivatives around the point $(k_0 = L/2, T_0 = n/2)$, which is near its global minimum. We have:

$$\begin{aligned} h(L/2,n/2) &= -\tfrac{nL^2}{4(\frac{L}{2})} + \tfrac{\frac{n}{2}(L-1)^2}{4(\frac{L}{2}-1)} + \tfrac{n}{2} + \tfrac{\frac{n}{2}(L-1)^2}{4(\frac{L}{2})} \\ &= \tfrac{n(L-1)}{2L(L-2)} \approx \tfrac{n}{2\lg n} \end{aligned}$$

$$\begin{aligned} \tfrac{\partial h}{\partial k}(L/2,n/2) &= +\tfrac{nL^2}{4(\frac{L}{2})^2} - \tfrac{\frac{n}{2}(L-1)^2}{4(\frac{L}{2}-1)^2} - \tfrac{\frac{n}{2}(L-1)^2}{4(\frac{L}{2})^2} \\ &= \tfrac{n(-3L^2+6L-2)}{L^2(L-2)^2} \approx \tfrac{-3n}{\lg^2 n} \end{aligned}$$

$$\begin{aligned} \tfrac{\partial h}{\partial T}(L/2,n/2) &= -\tfrac{\lg^2(n/2)+2(\lg e)\lg(n/2)}{4(\frac{L}{2}-1)} + 1 + \tfrac{\lg^2(n/2)+2(\lg e)\lg(n/2)}{4(\frac{L}{2})} \\ &= -\tfrac{(L-1)^2+2(L-1)\lg e}{4(\frac{L}{2}-1)(\frac{L}{2})} + 1 \\ &= \tfrac{-(L-1)^2+2(L-1)\lg e+L(L-2)}{L(L-2)} \\ &= \tfrac{2(L-1)\lg e-1}{L(L-2)} \approx \tfrac{2\lg e}{\lg n} \end{aligned}$$

$$\begin{aligned}\frac{\partial^2 h}{\partial k^2}(L/2, n/2) &= -\frac{n \lg^2 n}{2(\frac{L}{2})^3} + \frac{\frac{n}{2}\lg^2\frac{n}{2}}{2(\frac{L}{2}-1)^3} + \frac{\frac{n}{2}\lg^2\frac{n}{2}}{2(\frac{L}{2})^3} \\ &= \frac{-(\frac{L}{2}-1)^3 nL^2 + ((\frac{L}{2})^3 + (\frac{L}{2}-1)^3)\frac{n}{2}(L-1)^2}{2(\frac{L}{2})^3(\frac{L}{2}-1)^3} \\ &= \frac{n(4L^4+4L^3-44L^2+56L-16)}{L^3(L-2)^3} \approx \frac{4n}{\lg^2 n}\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 h}{\partial k \partial T}(L/2, n/2) &= \frac{\lg^2(n/2)+2(\lg e)\lg(n/2)}{4(\frac{L}{2}-1)^2} - \frac{\lg^2(n/2)+2(\lg e)\lg(n/2)}{4(\frac{L}{2})^2} \\ &= \frac{(4L-4)((L-1)^2+2(\lg e)(L-1))}{L^2(L-2)^2} \approx \frac{4}{\lg n}\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 h}{\partial T^2}(L/2, n/2) &= \frac{2(\lg e)(\lg e+\lg(n/2))/(n/2)}{4(\frac{L}{2}-1)} + \frac{2(\lg e)(\lg e+\lg(n/2))/(n/2)}{4(\frac{L}{2})} \\ &= \frac{(4L-4)(\lg e)(\lg e+(L-1))}{nL(L-2)} \approx \frac{4\lg e}{n}\end{aligned}$$

Using the first-order approximations of the first and second derivatives, we calculate that the function $h(k, T)$ will reach its global minimum at about

$$\begin{aligned}k &= k_0 + \frac{5\lg e}{4\lg e - 4} \approx \frac{\lg n}{2} + 4.07 \\ T &= T_0 + (\frac{-2\lg e - 3}{4\lg e - 4})\frac{n}{L} \approx \frac{n}{2} - 3.32\frac{n}{\lg n}\end{aligned}$$

and its value there will be

$$\frac{n}{2\lg n} - O(\frac{n}{\lg^2 n}),$$

which is positive for n sufficiently large. This concludes the proof.

References

1. G. Itkis and L. Reyzin, "Forward-Secure Signatures with Optimal Signing and Verifying," Crypto '01, pp. 332–354.
2. M. Jakobsson, "Fractal Hash Sequence Representation and Traversal," ISIT '02; full paper at `www.markus-jakobsson.com`.
3. M. Jakobsson "Method and Apparatus for Efficient Computation of One-Way Chains in Cryptographic Applications," U.S. Patent Application 09/969,833
4. L. Lamport, "Constructing Digital Signatures from a One Way Function," SRI International Technical Report CSL-98 (October 1979).
5. R. Merkle, "A digital signature based on a conventional encryption function," Proceedings of Crypto '87.
6. S. Micali, "Efficient Certificate Revocation," Proceedings of RSA '97, and U.S. Patent No. 5,666,416.
7. A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and Secure Source Authentication for Multicast," Proceedings of Network and Distributed System Security Symposium NDSS 2001, February 2001.
8. A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," Proc. of IEEE Security and Privacy Symposium S & P 2000, May 2000.
9. A. Perrig, R. Canetti, D. Song, and D. Tygar, "TESLA: Multicast Source Authentication Transform", Proposed IRTF draft, `http://paris.cs.berkeley.edu/~perrig/`

10. K. S. J. Pister, J. M. Kahn and B. E. Boser, "Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes. Highlight Article in 1999 Electronics Research Laboratory Research Summary.", 1999. See
`http://robotics.eecs.berkeley.edu/~pister/SmartDust/`
11. FIPS PUB 180-1, "Secure Hash Standard, SHA-1,"
`www.itl.nist.gov/fipspubs/fip180-1.htm`
12. S. Stubblebine and P. Syverson, "Fair On-line Auctions Without Special Trusted Parties," Financial Cryptography '01.

Cryptographic Primitives Enforcing Communication and Storage Complexity

Philippe Golle[1]*, Stanislaw Jarecki[2], and Ilya Mironov**[1]

[1] Stanford University, USA
[2] Intertrust, USA

Abstract. We introduce a new type of cryptographic primitives which enforce high communication or storage complexity. To evaluate these primitives on a random input, one has to engage in a protocol of high communication complexity, or one has to use a lot of storage. Therefore, the ability to compute these primitives constitutes a certain "proof of work," since the computing party is forced to contribute a lot of its communication or storage resources to this task. Such primitives can be used in applications which deal with non-malicious but selfishly resource-maximizing parties. For example, they can be useful in constructing peer-to-peer systems which are robust against so called "free riders." In this paper we define two such primitives, a communication-enforcing signature and a storage-enforcing commitment scheme, and we give constructions for both.

Keywords: Communication Complexity and Cryptography, Storage Complexity, Peer-to-Peer Networks

1 Introduction

Peer-to-peer networks have led to an explosive growth of file trading between Internet users. Much effort has been devoted to restricting this freewheeling market, on the assumption that Internet users are eager to trade as much as possible and should be legally prevented from doing so. However, peer-to-peer networks also suffer from the diametrically opposite problem of so-called "free riders," i.e. users who use the services provided by others but do not contribute anything to the network (e.g. see [AH00,SGG02]).

Motivated by this problem, we study the general question of how to construct efficiently verifiable proofs that a party in some protocol uses adequately high communication or storage resources. We propose a novel class of cryptographic primitives which enforce either high communication or high storage complexity on some party, where "high" means proportional to the size of some large input to the protocol. These primitives are interesting on theoretical grounds, because they constitute a unique application of cryptographic tools to enforce

* Supported by Stanford Graduate Fellowship.
** Supported by NSF contract #CCR-9732754.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 120–135, 2003.

linear lower-bounds on communication or storage resources. They can be used to generate a proof of work performed by some party, and are therefore applicable to settings that deal with non-malicious but selfish parties which try to minimize the use of their own resources. For example, they can be applied to auditing file trading or file storage in peer-to-peer networks and thus making such networks robust against free riders.

In this paper we define and give constructions for two new primitives:

- A *communication-enforcing signature (CES)*. CES is a signature scheme which enforces high communication complexity on the party that signs a message.
- A *storage-enforcing commitment (SEC)*. SEC is a commitment scheme which enforces high storage complexity on the party that commits to a message.

We explain both notions using the following two examples.

Consider an advertising agency that distributes digital ads (e.g., pictures or short videos) to viewers on behalf of advertisers. We assume that viewers are also encouraged to exchange ads among themselves. For example, a viewer might forward an interesting or amusing ad to a friend. Viewers must submit "proofs of download" to the ad agency, which rewards them in proportion to the number of ads they have downloaded. Observe that standard digital signatures are inadequate as proofs of download, since the *hash-and-sign* paradigm on which they are based implies that knowledge of the digest of a message is sufficient to sign the whole message. Viewers could save on bandwidth by exchanging only the hashes of ads rather than the ads themselves, and claim credit for ads they never downloaded. We propose instead to use communication-enforcing signatures (CES) as proofs of download. CES share the same security properties as standard digital signatures, but offer the added guarantee that, short of leaking his private-key, the signer can only sign a message after exchanging at least as much data with the source of the message as would be required to send the message. In our example, the ad agency could collect CES from viewers and submit them to advertisers for the purpose of auditing the success of an ad campaign.

We now consider another example, to illustrate this time the need for auditing file storage. Assume that a couple entrusts a public notary with their will. The couple does not want to reveal this will to their children, but at the same time they want their children to have the ability to verify that the public notary is storing their will. In essence, these requirements amount to a proof of storage of a secret document. In this situation, our *storage-enforcing commitment* (SEC) scheme could be used to audit file storage. Indeed, SEC allows the party holding a commitment to ask the party that committed itself to a message to prove that it is still in possession of this message, or, more precisely, that it uses at least as much storage as would be required to store the message. In our example, the notary would give to the children a storage-enforcing commitment to the will, which would enable the children to verify in the future that the notary is not saving on storage by erasing the will.

Organization. The rest of the paper is organized as follows. In the rest of this section, we give a brief survey of related work. In section 2, we give

a formal definition of communication-enforcing signatures (CES) and propose heuristic constructions for CES. We also show that the ability to compute on encrypted data would imply that CES are not possible. This leads us to believe that heuristic constructions for CES is the best we may hope for. In section 3, we define storage-enforcing commitments (SEC) and give a *provably secure* SEC construction. We conclude in section 4.

1.1 Related Work

Much work has been devoted to the study of communication complexity in multi-party computations. Mehlhorn et al. showed in [MS82] that there is a relationship between the communication complexity of a boolean function and the rank of the associated matrix. This relationship was also investigated in [Yao83,NW94].

However, there are surprisingly few results on communication complexity under cryptographic assumptions. A notable exception are digital signets introduced by Dwork et al. in [DLN96]. Signets are based on a primitive called incompressible functions. A length-increasing function f is incompressible if, in order to communicate $f(x)$ to Bob, Alice must either reveal her secret x to Bob, or else send Bob a message of length $\Omega(|f(x)|)$.

CES mirror the properties of incompressible functions in reverse: Alice may either send a short message to Bob (her private key) or *receive* from Bob the message to be signed. In [DLN96], the authors conjecture that some functions are incompressible, but leave as an open problem the task of building an incompressible function based on standard cryptographic primitives, or prove that such functions can not exist. Similarly, this paper offers only heuristics for building CES.

We propose a provably secure construction for a storage-enforcing commitment scheme, based conceptually upon Toeplitz matrices. These matrices have also been used for efficient constructions of a MAC [Kra94] and universal hash functions [MNT93].

2 Communication-Enforcing Signature Scheme

In this section we define communication-enforcing signatures (CES). We propose a few simple CES schemes with heuristic security. We observe that a signature scheme cannot be provably communication-enforcing if computation on encrypted data were feasible. From a theoretical point of view, this suggests that our heuristic constructions may be the best one can hope for. In practice, we believe that our constructions are good heuristics for enforcing communication complexity as long as there are no efficient methods for computing on encrypted data.

2.1 Definition of a CES

A CES scheme has all the properties of a regular signature scheme [GMR88], but in addition it is also "communication-enforcing". This means that in order

to sign a message held by a source S, a signer P must either enable S to sign any message on its behalf, or else engage with S in a protocol of communication complexity equal at least to the length of the message to be signed.

A stronger notion of communication-enforcement would require the message to be computable from the signature, perhaps by an all-powerful machine. However, such a notion could only be satisfied if the length of the signature was at least the length of the message. In contrast, our definition of CES allows for schemes in which the size of the signature is polynomial only in the security parameter, as in standard digital signature schemes. This weaker notion of communication-enforcement appears good enough for our purpose. It creates disincentives for the owner of a private key to avoid the communication complexity associated with downloading a message that it must sign. Indeed, short of revealing his private key, the signer is forced to download *some* file of size comparable to the message.

Definition 1. *A communication-enforcing signature scheme (CES) is a triple of probabilistic polynomial-time algorithms:*

- *A key-generation algorithm G which outputs a private/public key pair (d, e) for every security parameter k;*
- *A signing algorithm* Sig *which given a message M and a private key d computes a signature $sig = \mathsf{Sig}(M, d)$;*
- *A verification algorithm* Ver *s.t. if $(d, e) = G(k)$ and $sig = \mathsf{Sig}(M, d)$ then $\mathsf{Ver}(M, sig, e) = 1$;*

such that for every probabilistic polynomial-time interactive algorithm CompSig *between a signer P who has a private key d and a message source S who has a message M,*

- *If for every private key d and message M,* CompSig *outputs a valid signature sig on M,*
- *and if, after repeated executions of* CompSig *with P on polynomially-many messages $M_1, ..., M_n$ of his choice, S cannot forge a signature on some $M \neq M_1, ..., M_n$, except with probability negligible in the security parameter k*
- *then the communication complexity of the* CompSig *protocol for messages $M \in \{0,1\}^m$ is at least m.*

We can restate this definition in simpler terms as follows. No matter what protocol the signer P and the message source S engage in to produce the signature $\mathsf{Sig}(M, d)$ on message M, either the communication complexity of this protocol is at least the size of M, or the signing protocol enables the source to forge signatures. Thus the definition above incorporates the unforgeability notion of [GMR88].

2.2 Implications of Computing on Encrypted Data

It was pointed out in [San01] that if there exists a method for performing general computation on encrypted data then communication-enforcing signature

schemes cannot exist. The problem of computing on encrypted data was posed twenty years ago by Yao [Yao82]. If there existed an efficient solution to this problem, a CES scheme could not exist for the following reason. The signer P could send an *encryption* of its private (signing) key to the source S. S would in turn encrypt the message M and evaluate the function Sig on encrypted inputs to compute an encryption of the output $\sigma = \mathsf{Sig}(M, d)$. This (short) output could then be sent back to P, who can decrypt it to recover the signature σ. Thus P could compute σ without revealing its inputs to S, and the communication complexity of this protocol would be independent of $|M|$.

Since there is nothing to suggest that it is impossible to compute on encrypted data, we conclude that we cannot hope to prove that a signature scheme is communication-enforcing. The constructions we propose next are only *heuristically* communication-enforcing.

2.3 First Heuristic Construction for CES: CBC-RSA

A simple implementation of a CES would be to apply a trapdoor permutation to each block of the message in turn. However, such a scheme is impractical because the size of the resulting signature is as long as the input message.

To reduce the output size, we may use a CBC-chain as follows. Assume that we are using the RSA function [RSA77] with k-bit modulus and a hash function H with k-bit long output size, and that the message M is divided into blocks $M_1 ... M_n$ each of size k. The prover signs the first block $C_1 = H(M_1)^d \bmod N$, and for each i-th block for $i = 2, ..., n$, he computes $C_i = (C_{i-1} \oplus H(M_i))^d \bmod N$, and so on until $C_n = (C_{n-1} \oplus H(M_n))^d \bmod N$. The resulting communication-enforcing signature is equal to the last block signature C_n. Since the block-wise RSA signature operation we use supports message-recovery, the verifier algorithm can unwind this CBC construction as follows. He computes $C_{i-1} = (C_i^e \bmod N) \oplus H(M_i)$ for every $i = n, ..., 2$ until it recovers C_1, then verifies that $C_1^d \bmod N = H(M_1)$.

This CBC-chaining construction enforces high communication complexity if we model the private-key RSA permutation as a random oracle which can only be accessed by the party who knows the private key.

The drawback of our CBC-chaining construction is that it relies on a heuristic security argument and has a high computational cost. Indeed the signer needs to perform one private-key operation per block of the message M, where each block is about 1024 bits if we use 1024-bit RSA. This implies that a signer would need to perform some 100,000 RSA signature operations in order to sign a 10 MBytes file.

2.4 Another Heuristic Construction for CES: Two-Root RSA

In this section we present a signature scheme, which is secure in the random oracle model and heuristically communication-enforcing. The scheme is based on a generalization of the RSA signature scheme [RSA77].

Consider the following typical implementation of the RSA signature scheme. A signature on a message M is a solution of the following equation in $\mathbb{Z}_N$:

$$x^3 = H(M) \pmod{N},$$

where H is a collision-resistant hash function and N is a product of two primes.

We generalize this scheme to include more blocks of the message as follows. A signature on a message $M = (M_0, \ldots, M_n)$ is a pair of two distinct roots $x_1, x_2 \in \mathbb{Z}_N$ of the following polynomial in $\mathbb{Z}_N[x]$:

$$H(M_n)x^n + H(M_{n-1})x^{n-1} + \cdots + H(M_1)x + H(M_0) = 0 \pmod{N}, \quad (1)$$

where H is a full-domain length-preserving collision-resistant function and $N = pq$ is a product of two prime numbers. The roots x_1, x_2 are subject to the condition that $x_1 - x_2$ is coprime with N.

Using the Ben-Or algorithm [Ben81] a root of this polynomial, if it exists, can be found in time $O(n \log n \log\log n \log^2 N)$. By the Chinese Remainder Theorem the polynomial has two roots $x_1, x_2 \in \mathbb{Z}_N$ such that $\gcd(x_1 - x_2, N) = 1$ if and only if it has at least two distinct roots both in $\mathbb{Z}_p$ and $\mathbb{Z}_q$. The existence of at least two roots in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are independent events, each with probability between $1/3$ and $1/2$ [Knu75, section 4.6.2]. Therefore a solution to equation (1) exists with probability between $1/9$ and $1/4$. As n increases the probability approaches $e^{-2} \approx .135$.

We apply this concept to design a signature scheme which is as secure as factoring (in the random oracle model). This is done by introducing chaining between blocks, as follows.

Signing algorithm. The message M is formatted as $(M_1, \ldots, M_n)$, $n > 1$. The following algorithm is executed by the signer until step 5 succeeds (the expected number of attempts required depends on n and is between 4 and 9).

Step 1. Randomly choose an initial vector $IV \xleftarrow{R} \mathbb{Z}_N^*$.
Step 2. Compute $C \leftarrow H(M_1, H(M_2, \ldots H(H(0, IV), M_n) \ldots)$.
Step 3. Compute $C_i \leftarrow H(C, M_{i+1})$ for $0 \le i < n$.
Step 4. Define $P(x) \leftarrow x^n + C_{n-1}x^{n-1} + C_{n-2}x^{n-2} + \cdots + C_1 x + C_0$.
Step 5. Find two distinct roots $t_1, t_2 \in \mathbb{Z}_p$ and two distinct roots $s_1, s_2 \in \mathbb{Z}_q$ of $P(x)$.
Step 6. Find $x_1, x_2 \in \mathbb{Z}_N$ satisfying

$$x_1 \equiv t_1 \pmod{p} \qquad x_1 \equiv s_1 \pmod{q}$$
$$x_2 \equiv t_2 \pmod{p} \qquad x_2 \equiv s_2 \pmod{q}.$$

Step 7. Output the signature (IV, x_1, x_2).

Verification algorithm. A signature on a message $M = (M_1, \ldots, M_n)$ is parsed as (IV, x_1, x_2). The verifier computes $C, C_1, \ldots, C_n$ and $P(x) \in \mathbb{Z}_N[x]$ as above and checks that $x_1 \not\equiv x_2 \pmod{N}$. The signature is accepted if $P(x_1) = 0 \pmod{N}$ and $P(x_2) = 0 \pmod{N}$.

In the random oracle model this signature scheme is existentially unforgeable against adaptive chosen message attacks assuming the hardness of factoring. We

omit the standard part of the argument in the random oracle model (see [BR93] for an example of a detailed proof) and sketch the reduction from the ability to find two roots of a random polynomial of degree n in $\mathbb{Z}_N[x]$ to factoring N.

Claim. The problem of finding two distinct roots of a random polynomial of degree n drawn from the uniform distribution on $\mathbb{Z}_N[x]$ is as difficult as factoring.

Proof Sketch: Suppose there exists an algorithm $\mathcal{A}$ that given $P(x) \stackrel{R}{\leftarrow} \mathbb{Z}_N[x]$ of degree n outputs two distinct roots (x_1, x_2) of $P(x)$ with non-negligible probability ε. We build an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to factor N as follows:

Step 1. Choose $y_1, y_2 \stackrel{R}{\leftarrow} \mathbb{Z}_N$.
Step 2. Choose $Q(x) \stackrel{R}{\leftarrow} \mathbb{Z}_N[x]$ of degree $n-2$.
Step 3. Compute $P(x) = (x - y_1)(x - y_2)Q(x)$.
Step 4. Run $\mathcal{A}$ on $P(x)$. Let the output of $\mathcal{A}$ be (x_1, x_2).
Step 5. If either of $x_1 - y_1, x_1 - y_2, x_2 - y_1, x_2 - y_2$ is not zero and not coprime with N, we have found a nontrivial factor of N.

The distribution from which Q is drawn in steps 1–2 is off by at most a factor of n^2 from the uniform distribution on polynomials of degree n with at least two roots. Therefore, the probability of success in step 4 is at least ε/n^2.

Notice that along with known roots y_1, y_2 of $Q(x)$ there exist two unknown roots (y_1', y_2') satisfying

$$y_1' \equiv y_1 \pmod{p} \qquad y_1' \equiv y_2 \pmod{q}$$
$$y_2' \equiv y_2 \pmod{p} \qquad y_2' \equiv y_1 \pmod{q}.$$

Any one of y_1', y_2' reveals the factorization of N. The probability that $\mathcal{A}$ outputs one of these two values is no less than $\frac{1}{n^2-2}$. Therefore $\mathcal{B}$ succeeds in factoring N with probability at least ε/n^4. □

It is interesting to note that it is not known whether a similar signature scheme in which the signature of a message consists of a single root of a polynomial from $\mathbb{Z}_N[x]$ is secure under any standard assumption.

Heuristically this scheme is communication-enforcing, since it is believed to be hard to find a root of a polynomial without full knowledge of this polynomial.

3 Storage-Enforcing Commitment

In this section we introduce a primitive called *storage-enforcing commitment scheme*. Its setup is similar to commitment schemes, but the scheme has the additional property that the committer (whom we call the prover) cannot discard the secret it is supposed to store. This problem is trivial if the storage complexity of the verifier or the complexity of its communications with the prover is unbounded. However, we are able to propose a practical storage-enforcing commitment scheme, for which the storage and the amortized communication complexity are independent of the length of the message.

Regular commitment schemes [Blu83] bind the prover to a particular value of a string that is to be kept secret during some stages of the execution of a protocol.

These commitment schemes are not designed to permit repeated verification of a commitment to the same string. Storage-enforcing commitment schemes on the other hand are multi-round protocols that ensure that the prover neither "forgets" nor alters the secret between rounds. Any prover who is able to answer the verifier's challenges must keep the secret, or at least use as much storage space as would be required to store the secret. Formally, we define:

Definition 2 (Storage-enforcing commitment scheme). *A* storage-enforcing commitment scheme *(SEC) is a three-party protocol executed between a message source S, a prover P, and a verifier V. The message source communicates the message M to the prover and the commitment C to the verifier. The verifier V may verify whether the prover is storing the secret by invoking a probabilistic interactive algorithm* $\mathsf{Check}_{P,V}(C, M)$. *This algorithm may be executed an unlimited number of times. Once the message is revealed, the verifier may check the commitment by running the algorithm* $\mathsf{Verify}(C, M)$.

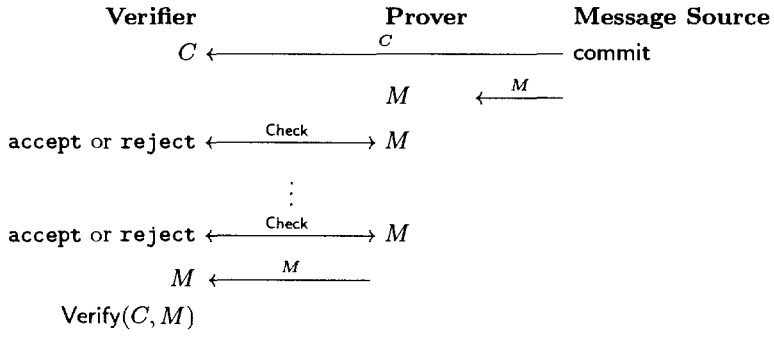
The scheme has the following three properties called binding, concealing, *and* storage-enforcing*:*

Binding. *A coalition of a computationally bounded message source and a prover cannot find two strings M and M' that both pass the test* Verify *with the same commitment C.*

Concealing. *A computationally unbounded verifier V cannot decide if $M = M_1$ or $M = M_2$ prior to the opening of the commitment with non-negligible advantage over a random guess even if M is known to be either M_1 or M_2 and the strings M_1 and M_2 were chosen by V.*

Storage-Enforcing. *Any prover $\hat{P}$ that passes the test* Check *with probability $\alpha > 0$ has storage complexity $\Omega(|M|)$. The probability is taken over all messages of a fixed length $|M|$ and over the coin tosses of the prover and the verifier.*

Table 1. Storage-enforcing commitment scheme



To illustrate this definition, recall our example from the introduction. A couple entrusts a public notary with their will. The couple doesn't want their

children to learn the will, yet the children should be able to verify that the public notary is properly storing the will. These requirements are satisfied by a storage-enforcing commitment scheme, where the notary is the prover and the children are verifiers. The verification protocol enables the children to verify that the notary is still in possession of the will (or at least, that the notary uses at least as much storage space as would be required to store the will), yet it does not leak any information about the will.

Unsatisfactory solutions. We explain here why other possible commitment schemes do not meet the requirements of our example. Consider first that we could have the message source send to the verifier a non-compressing commitment to each block of the message. The verifier stores hashes of all these commitments. To execute the protocol Check, the verifier requests from the prover a commitment to a random block, hashes it and compares it to the hash previously stored. The problem with this approach is that it requires the verifier to store an amount of data (in the form of commitments) which is proportional to the amount of data the prover has to store. This defeats the point of having the prover store the data on behalf of the verifier.

Alternately, instead of storing hashes of all the commitments, the verifier could store a single hash computed on all the commitments concatenated. This approach however requires the prover to send *all* the commitments each time the protocol Check is executed. This leads to a scheme with unacceptable communication complexity.

3.1 Our Storage-Enforcing Commitment Scheme

We work in a group G of prime order p with generator g. The order of the group depends on the security parameter λ. Our scheme is based on a variant of the Decisional Diffie-Hellman assumption, which we call the n-Power Decisional Diffie-Hellman (n-PDDH) assumption.

n-Power Decisional Diffie-Hellman (n-PDDH) assumption. No polynomial-time algorithm can distinguish between the following two distributions with non-negligible advantage over a random guess:

Distribution $\mathcal{P}^n$: $(g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^n})$, where $x \stackrel{R}{\leftarrow} \mathbb{Z}_p$,

and

Distribution $\mathcal{R}^n$: $(g_1, g_2, \ldots, g_n)$ where $g_1, g_2, \ldots, g_n \stackrel{R}{\leftarrow} G$.

Notice that 2-PDDH is the same as the Decisional Square Exponent Diffie-Hellman assumption [BDS98,SS01]. We also need a weaker, computational assumption defined below.

n-Power Computational Diffie-Hellman (n-PCDH) assumption. No probabilistic polynomial-time algorithm can compute g^{x^n} given $g^x, g^{x^2}, \ldots, g^{x^{n-1}}$ with non-negligible probability.

The 2-PCDH assumption is equivalent to the Computational Diffie-Hellman assumption. It is unknown whether n-PCDH implies $(n+1)$-PCDH (the converse is obviously true).

Design of the scheme. We limit the size of a message to m blocks, where blocks are interpreted as elements of $\mathbb{Z}_p$. Longer messages can be broken into pieces and committed to simultaneously. Let $n = 2m + 1$ and assume that n-PDDH holds in the group G.

The **secret key** of the verifier is x randomly chosen from $\mathbb{Z}_p$. The corresponding **public key** is

$$PK = (g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^n}) = (g_1, \ldots, g_n).$$

The *commitment phase* consists of the following three steps:

Step 1. The verifier publishes the public key PK.

Step 2. The verifier gives a zero-knowledge proof to P that the key is properly constructed, i.e. for any index $1 \leq i < n$ the quadruple (g, g_1, g_i, g_{i+1}) is a DH-tuple [CP92].

Step 3. The message source S formats the message as an m-tuple $M = (M_1, \ldots, M_m) \in \mathbb{Z}_p^m$. S chooses a random element $M_{m+1} \in \mathbb{Z}_p^*$ and appends it to the message.

Step 4. S computes $f_0 = \prod_{i=1}^{m+1} g_i^{M_i}$ and sends it to the verifier. $f_0 \in G$ is the commitment to the message.

From the verifier's point of view the blinding block M_{m+1} is considered as an integral part of the message. For the purpose of computing the storage complexity we include this block into the message.

To check that the prover still has the message, the verifier initiates the following Check protocol:

Step 1. The verifier sends to the prover a challenge $0 \leq k \leq m$.

Step 2. The prover computes $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i}$, where M_i is the i^{th} block of the message and sends f_k back to the verifier.

Step 3. The verifier checks whether $f_0^{x^k} = f_k$. If the equality holds, the verifier accepts the proof, otherwise the proof is rejected.

The *verification phase* is trivial—given the message $\hat{M}$, the verifier computes $\hat{f}_0$ and compares it to the f_0 received from the message source during the commitment phase.

The storage complexity of the verifier is one group element f_0 and $x \in \mathbb{Z}_p$. Since the verifier's public key can be reused for multiple messages, the scheme's storage overhead approaches zero on a per-message basis.

3.2 Proof of This Commitment Scheme

The commitment scheme described above is unconditionally concealing and satisfies the binding property against a computationally-bounded adversary under the n-PCDH assumption. Our proofs are generalizations of [Ped91].

Computational binding. Suppose that the message source can find two messages that may be committed to the same string C. This means that, for a given public key $g_1, \ldots, g_n$, S can find two messages $M = (M_1, \ldots, M_m)$ and $M' = (M'_1, \ldots, M'_{m'})$ such that

$$g_1^{M_1} g_2^{M_2} \ldots g_m^{M_m} = g_1^{M'_1} g_2^{M'_2} \ldots g_{m'}^{M'_{m'}}.$$

It follows that the following equation is satisfied:

$$M_1 x + M_2 x^2 + \cdots + M_m x^m = M'_1 x + M'_2 x^2 \cdots + M'_{m'} x^{m'} \pmod{p}.$$

Since $M_m, M_{m'} \neq 0 \pmod{p}$ and $M \neq M'$, the two sides of the equation are not identical. Therefore the equation can be solved for x in quasi-linear time [Ben81]. This violates the n-PCDH assumption.

Perfect concealing. The concealing property holds because the only information the verifier learns about the message prior to the opening phase is determined by the value of $\sum_{i=1}^{m+1} M_i x^i \bmod p$, which is independent of the message due to the blinding block M_{m+1}. This property is unconditional and holds against a computationally unbounded verifier.

Storage-enforcing. The following theorem proves that the scheme is storage-enforcing.

Theorem 1. *Any prover $\hat{P}$ that passes the test* Check *with probability $\alpha > 0$ has storage complexity at least $\alpha|M|$ under the n-PDDH assumption. The probability is taken over messages drawn from the uniform distribution on $\{0,1\}^{|M|}$, the public key of the verifier, and all coin tosses.*

Proof. We present the proof in two parts. First, we show (under the n-PDDH assumption) that the storage complexity of the prover and his probability of success are independent on whether the public key is chosen from $\mathcal{R}^n$ or $\mathcal{P}^n$. Second, we demonstrate that in order to pass the test with probability α when the key comes from $\mathcal{R}^n$, the prover must have storage complexity at least $\alpha|M|$. The second part of the proof is information-theoretic and unconditional.

We model the prover at different points in time as two separate algorithms $\hat{P}_1$ and $\hat{P}_2$. These two algorithms communicate by means of a bit string output by $\hat{P}_1$ and received by $\hat{P}_2$. The length of this string is at least the storage complexity of the prover.

Suppose we are given an instance of the n-PDDH problem: we must decide whether the tuple $(g_1, \ldots, g_n) \in G^n$ belongs to $\mathcal{P}^n$ or $\mathcal{R}^n$. The claim is that a prover with storage complexity less than $\alpha|M|$ can be used as a distinguisher between these two distributions. We set the public key of the verifier to be the n-tuple $(g_1, \ldots, g_n)$ and simulate the prover's view such that the only difference between this game and a real execution of the scheme is the type of the tuple.

The zero-knowledge proof of the key's correctness can be simulated using standard techniques. We assume that the transcript of the proof is given to $\hat{P}$.

Let the message M be a random string consisting of $m+1$ blocks. The message is known and therefore we can check the responses of $\hat{P}$ by a direct computation of $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i}$ without knowing x or even without being aware of whether the public key has the right structure. This simulates perfectly the prover's view *and* enables us to check his output.

If the prover has probability α of passing the test when the public key is constructed correctly but has a different probability of success when the key is a random n-tuple, this prover can be used as a distinguisher between the two

distributions. Therefore his probability of success in either case is negligibly close to α.

Similarly, the storage complexity, modelled as the communication complexity between $\hat{P}_1$ and $\hat{P}_2$, is an observable characteristic. Therefore it must be the same for the keys drawn from $\mathcal{R}^n$ and $\mathcal{P}^n$. Otherwise we could use the observable difference between the storage complexity to break the n-PDDH assumption.

We have just proved that $\hat{P}$ must be able to pass the test Check with the same probability and storage complexity regardless of whether PK is properly constructed or drawn from $PK \stackrel{R}{\leftarrow} \mathcal{R}^n$.

We now show that in order to pass the test with a key chosen from $\mathcal{R}^n$ the prover must have storage complexity at least $\alpha|M|$, where α is the probability of success.

Consider the following random tuple: $g_1 = g^{a_1}, \ldots, g_n = g^{a_n}$. To pass the test the prover must compute $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i} = \prod_{i=1}^{m+1} g^{a_{i+k} M_i}$ for a random k. We claim that the vector $(f_0, \ldots, f_m)$ has guessing entropy $|M|$. For $\hat{P}_2$ to reproduce a value from this list with probability at least α its input must be at least $\alpha|M|$ bit long.

Let the discrete logarithms of $(f_0, \ldots, f_m)$ to the base g be $(b_0, \ldots, b_m)$. It is easy to see that

$$b_i = (a_i, \ldots, a_{i+m}) \cdot (M_1, \ldots, M_{m+1}),$$

where $\cdot$ denotes the scalar product of two vectors from $\mathbb{Z}_p^n$. Therefore the entire vector $(b_0, \ldots, b_m)$ can be computed as

$$\begin{pmatrix} a_1 & a_2 & \ldots & a_m & a_{m+1} \\ a_2 & a_3 & \ldots & a_{m+1} & a_{m+2} \\ & & \ddots & & \\ a_m & a_{m+1} & \ldots & a_{n-2} & a_{n-1} \\ a_{m+1} & a_{m+2} & \ldots & a_{n-1} & a_n \end{pmatrix} \cdot \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_m \\ M_{m+1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \\ b_m \end{pmatrix}.$$

Let us denote the matrix on the left-hand side of this equation by A. This is a Toeplitz matrix defined by the values $(a_1, \ldots, a_n)$. The following lemma proves that this matrix is non-singular with overwhelming probability[1].

Lemma 1. *The Toeplitz matrix A defined above has full rank with probability at least $1 - \frac{n}{p}$ over all tuples $(a_1, \ldots, a_n)$ randomly chosen from $\mathbb{Z}_p^n$.*

Proof. We use the Schwartz Lemma [Sch80], which can be briefly stated as follows:

> Consider a non-trivial multivariate polynomial of degree n defined on variables from $\mathbb{Z}_p$. Given a random assignment to the variables, this polynomial evaluates to zero with probability at most $1 - n/p$.

[1] Using more advanced techniques [KL96] showed that a Toeplitz matrix is non-singular with probability $1 - 1/p$.

The determinant of A is a polynomial of n variables $(a_1, \ldots, a_n)$. This polynomial is not identically zero, since the matrix defined by $a_m = 1$ and $a_i = 0$ for all other $i \neq m$ has determinant $(-1)^{n-1}$. Therefore, by the Schwartz Lemma, the determinant of the Toeplitz matrix with randomly chosen entries is non-zero with probability at least $1 - n/p$. □(Lemma)

When the matrix A is full-rank, the vector $(b_0, \ldots, b_m)$ can take any value from $\mathbb{Z}_p^{m+1}$ independently of the public key. Since we assume that messages are uniformly distributed, the resulting distributions of $(b_0, \ldots, b_m)$ and the prover's responses $(f_0, \ldots, f_m)$ are also uniform. In this case the guessing entropy is equal to the Shannon entropy, which is $|M|$ bits. Therefore the prover must possess at least $\alpha|M|$ bits to give a correct response with probability α. This completes the proof of the theorem. □

3.3 Heuristic Extensions

Apart from two non-classical assumptions (n-PDDH and n-PCDH), the scheme presented in the previous section is provably secure in the standard model of computation. We propose here various heuristic extensions to make the scheme more practical.

Random oracles. Recall that the first step of Commit is a zero-knowledge proof given by V that the key is properly constructed. We can make this step non-interactive by applying the Fiat-Shamir heuristic [FS87,BR93] to the 3-round Chaum-Pedersen protocol. This non-interactive proof can be part of the verifier's public key. In the random oracle model we can attach this proof even to a random tuple, thus preserving the validity of the n-PDDH assumption.

DDH-oracle. We may conjecture that our scheme is secure under a weaker assumption than the decisional variant of the Diffie-Hellman assumption. Assume that there exists a group in which there is a DDH-oracle that tests a tuple (g, g^a, h, h^b) for the equality $a = b$ but in which the computational n-PCDH problem is hard. The binding and concealing properties of the scheme hold and it is not unreasonable to believe that the scheme may still be storage-enforcing even though the proof of Theorem 1 is not applicable here. In this case additional optimizations of the scheme are possible. We briefly sketch the resulting scheme:

With a DDH-oracle, the main improvement to the scheme comes from the fact that all receivers can share the *same* public-key:

$$(g_1, \ldots, g_n) = (g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^n}).$$

This **common parameter** may be generated once and for all during a set-up phase by a trusted authority. The value x used to generate this common parameter can then be discarded and the scheme requires no further involvement of the trusted authority.

The **commitment** is computed as follows:

Step 1. The message source S formats the message as an m-tuple $M = (M_1, \ldots, M_m) \in \mathbb{Z}_p^m$. S chooses a random element $M_{m+1} \in \mathbb{Z}_p^*$ and appends it to the message.

Step 2. S computes $f_0 = \prod_{i=1}^{m+1} g_i^{M_i}$ and sends it to the verifier. $f_0 \in G$ is the commitment to the message.

The Check protocol queries the DDH-oracle, which we denote as DDH:

Step 1. The verifier sends to the prover a challenge $0 \leq k \leq m$.

Step 2. The prover computes $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i}$ and sends it back to the verifier.

Step 3. The verifier queries $\mathsf{DDH}(g, g_k, f_0, f_k)$. If the tuple is a Diffie-Hellman tuple, the proof is accepted.

The verification of the revealed message is trivial: the verifier recomputes f_0 from the message.

Since in this case the verifier does not have secret data, the scheme can be used in a scenario where the verifier outsources his entire storage to the prover (the public key and the commitment must be signed and come with certificates).

This scheme is provably secure in the generic algorithm model [Sho97]. This is a weaker security argument, but for many practical protocols it is often the only guarantee we have (see [Sch00] for a survey of this situation).

The first example of a group in which the Decisional Diffie-Hellman problem is easy while its computational counterpart is hard was recently proposed by Joux and Nguyen [JN01]. It is a group of points on an elliptic curve of a special type, in which the Weil pairing is non-trivial and efficiently computable. We refer the interested reader to [BF01] for details on how to build a practical cryptographic scheme from the Weil pairing.

4 Conclusion and Further Work

We have introduced cryptographic primitives enforcing communication and storage complexity and proposed constructions for them. A general direction for future work would be to further investigate the connections between communication complexity and cryptographic assumptions.

Acknowledgements. The authors would like to thank Dan Boneh, Moni Naor, Ron Rivest, Tomas Sander, and Victor Shoup for many helpful discussions on the subject of this paper. We are also indebted to Vitaly Shmatikov for introducing us to one another.

References

[AH00] E. Adar and B. Huberman, "Free Riding on Gnutella," First Monday, 5(10), 2000

[BG92] M. Bellare and O. Goldreich, "On defining proofs of knowledge," Proc. of CRYPTO'92, pp. 390–420, 1992.

[BKR94] M. Bellare, J. Killian, and P. Rogaway, "The security of the cipher block chaining message authentication code," Proc. of CRYPTO'94, pp. 341–358.http://www.cs.ucdavis.edu/~rogaway, 1994.

[BR93] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," Proc. of ACM CCS'93, pp. 62–73, 1993.

[Ben81] M. Ben-Or, "Probabilistic algorithms in finite fields," Proc. of FOCS'81, pp. 394–398, 1981.

[Blu83] M. Blum, "Coin flipping by telephone," Proc. of CRYPTO'81, pp. 11–15, 1981.

[BF01] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," Proc. of CRYPTO'01, pp. 213–229, 2001.

[BDS98] M. Burmester, Y. Desmedt, and J. Seberry, "Equitable key escrow with limited time span (or, How to enforce time expiration cryptographically)," Proc. of Asiacrypt'98, pp. 380–391, 1998.

[CCKM00] C. Cachin, J. Camenish, J. Kilian, and J. Muller, "One-round secure computation and secure autonomous mobile agents," Proc. of ICALP'00, pp. 512–523, 2001.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology revisited," Proc. of STOC'98, pp. 209–218, 1998.

[CP92] D. Chaum and T. Pedersen, "Wallet databases with observers," Proc. of CRYPTO'92, pp. 89–105, 1992.

[DLN96] C. Dwork, J. Lotspiech, and M. Naor, "Digital signets: self-enforcing protection of digital content," Proc. of STOC'96, pp. 489–498, 1996.

[FS87] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," Proc. of CRYPTO'86, pp. 186–194, 1987.

[Gol98] O. Goldreich, "Secure multi-party computation," On-line manuscript, http://www.wisdom.weizmann.ac.il/~oded, 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," Proc. of STOC'87, pp. 218–229, 1987. (See also [Gol98]).

[GMR88] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM J. on Computing, 17(2), pp. 281–308, 1988.

[JN01] A. Joux and K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups," Cryptology ePrint Archive, Report 2001/003, available form http://eprint.iacr.org/, 2001.

[KL96] E. Kaltofen and A. Lobo, "On rank properties of Toeplitz matrices over finite fields," Proc. of ISSAC'96, pp. 241–249, 1996.

[Knu75] D. Knuth, *The Art of Computer Programming, v. 2, Seminumerical Algorithms*, 2nd Ed., Addison-Wesley, 1975.

[Kra94] H. Krawczyk, "LFSR-based hashing and authentication," In Proc. of CRYPTO'94, pp. 129–139, 1994.

[MNT93] Y. Mansour, N. Nisan, and P. Tiwari, "The computational complexity of universal hash functions," Theoretical Computer Science, v. 107(1), pp. 121–133, 1993.

[MS82] K. Mehlhorn and E. Schmidt, "Las Vegas is better than determinism in VLSI and distributed computing," Proc. of STOC'82, pp. 330–337, 1982.

[NW94] N. Nisan and A. Wigderson, "On rank vs. communication complexity," Proc. of FOCS'94, pp. 841–836, 1994.

[Ped91] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," Proc. of CRYPTO'91, pp. 129–140, 1991.

[RSA77] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Comm. of ACM, 21, pp. 120–126, 1977.

[SS01] A. Sadeghi and M. Steiner, "Assumptions related to discrete logarithms: why subtleties make a real difference," Proc. of EUROCRYPT'01, pp. 244–261, 2001.

[San01] T. Sander, private communications, 2001.

[SGG02] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," Proc. of Multimedia Computing and Networking 2002, January, 2002.

[Sch00] C. Schnorr, "Security of DL-encryption and signatures against generic attacks—A survey," Proc. of PKC&CNTC'2000, 2000.

[Sch80] J. Schwartz, "Probabilistic algorithms for verification of polynomial identities," J. of ACM, v. 27, pp. 701–717, 1980.

[Sho97] V. Shoup, "Lower bounds for discrete logarithms and related problems," Proc. of Eurocrypt'97, pp. 256–266, 1997.

[Yao82] A.-C. Yao, "Protocols for secure computations," Proc. of FOCS'82, pp. 160–164, 1982.

[Yao83] A.-C. Yao, "Lower bounds by probabilistic arguments," Proc. of FOCS'83, pp. 420-428, 1983.

CryptoComputing with Rationals

Pierre-Alain Fouque[1,2], Jacques Stern[2], and Geert-Jan Wackers[3]

[1] D.C.S.S.I. Crypto Lab
51, bd Latour-Maubourg, F-75007 Paris, France
[2] École Normale Supérieure, Département d'Informatique
45, rue d'Ulm, F-75230 Paris Cedex 05, France
{Pierre-Alain.Fouque,Jacques.Stern}@ens.fr
[3] Ernst & Young EDP Audit,
The Netherlands
nlwacke1@ey.nl

Abstract. In this paper we describe a method to compute with encrypted *rational* numbers. It is well-known that *homomorphic schemes* allow calculations with hidden *integers*, *i.e.* given integers x and y encrypted in $\mathcal{E}(x)$ and $\mathcal{E}(y)$, one can compute the encrypted sum $\mathcal{E}(x+y)$ or the encrypted product $\mathcal{E}(kx)$ of the encrypted integer x and a known integer k without having to decrypt the terms $\mathcal{E}(x)$ or $\mathcal{E}(y)$. Such cryptosystems have a lot of applications in electronic voting schemes, lottery or in multiparty computation since they allow to keep the privacy of the terms and return the result in encrypted form. However, from a practical point of view, it might be interesting to compute with *rationals*. For instance, a lot of financial applications require algorithms to compute with rational values instead of integers such as bank accounts, electronic purses in order to make payments or micropayments, or secure spreadsheets. We present here a way to solve this problem using the Paillier cryptosystem which offers the largest bandwidth among all homomorphic schemes. The method uses two-dimensional lattices to recover the numerator and denominator of the rationals. Finally we implement this technique and our results in order to build an encrypted spreadsheet showing the practical possibilities of the homomorphic properties applied on rationals.

1 Introduction

A lot of financial applications use calculations such as sums of expenses, or multiplications by a public constant (rates). It is easy to show that electronic purses or bank accounts have to be encrypted from the users or clients point of view whereas some values can be public such as interest rates in order to update accounts at the end of each month. Therefore, we need to add or subtract two encrypted expenses (rational numbers) or to multiply encrypted numbers by a public rational. For example, if a bank wants to keep secret the accounts of its clients to the bank employees, a safe way consists in encrypting the bank accounts. There are two problems to solve : the first challenge is to encode

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 136–146, 2003.

rationals r such that the homomorphic properties of the cryptosystems still hold, and the second challenge is to reconstruct the numerator and denominator of the encrypted rational when recovery of the rationals is needed.

The basic idea to reduce the problem to compute with integers does not work. Indeed, one can think to multiply each value by 10^5 before encrypting them, provided we allow a precision of 10^{-5}. In this case, one can perform additions but *not* multiplications by a scalar. Consequently, one must use an encoding that respect both the multiplication and addition.

We can also remark that the encoding of the numerator and the denominator does not provide a valid solution. Indeed, let us consider the following encoding of the rational a/b as $[\mathcal{E}(a), \mathcal{E}(b)]$. If we have a multiplicative homomorphic scheme as the plain-RSA, then the multiplication of two rationals is a trivial task. However, the addition of two rationals is not possible, and furthermore, the addition is the more useful operation in spreadsheet.

1.1 Related Works

From a theoretical point of view, it is important to securely compute with encrypted data. Computing with hidden values or securely doing mathematical operations such as comparison of two encrypted numbers are deep operations in cryptography. The latter is known as the millionaire problem. It has been extensively studied to build efficient auction protocols. For example, at the last Financial Crypto, Baudron and Stern [3] have devised an algorithm to solve this problem. The problem is the following : given a set of encrypted bids $\{\mathcal{E}(v_i)\}_i$, find $\max(\mathcal{E}(v_i))$ without decrypting the bids. The important point of the Baudron and Stern algorithm is that the calculation is done without interactions between the participants.

The former problem of "computing with encrypted data" has been studied by numerous authors in the past twenty years. Sander, Young and Yung have proposed in [20] a protocol that non-interactively computes any function in $\mathcal{NC}^1$. This is an important result as it is an open problem to reduce the round of computation in general multiparty computation protocols. It is a particular case of the more general classical problem : Alice has an input x and Bob has circuit C. Alice should known the value $C(x)$ but nothing else "substantial" about C. Bob should learn nothing else "substantial" about Alice's input x. In our case, the function C can be learnt by anyone. Several papers more extensively discuss secure circuit evaluation such as [1].

Informally a cryptosystem is algebraically homomorphic if given the encryption of two plaintexts x and y, one can construct the encryption of the plaintexts $x+y$ and xy in polynomial time without revealing x or y. The existence of these schemes has been left open. In 1978, Rivest, Adleman, and Dertouzos [19] suggested investigating encryption schemes with additional homomorphic properties since they allow to compute with encrypted data. In 1991, Feigenbaum and Merritt [11] directly addressed algebraic homomorphic schemes of the form stated above. In [5], Boneh and Lipton showed that *deterministic* algebraically homomorphic encryption schemes over ring $\mathbb{Z}_N$ can be broken in subexponential

time under a (reasonable) number theoretic assumption. In their argument, it is essential though, that the scheme be deterministic.

Homomorphic cryptosystems provide an efficient way to solve the "Secure Function Evaluation" Problem for some mathematical operations. In particular, we focus here on, $(\times, +)$-homomorphic schemes such as [4,15,16,17], which are probabilistic and enable to perform addition of two encrypted values or multiplication by a known integer. Homomorphic schemes were proposed to solve more efficiently the problem of computing with integers under computational assumptions. These special kind of public-key cryptosystems have applications in electronic schemes or lotteries [4,9,12,2]. These algorithms can also be used to build efficient multiparty computation protocols with reduced communication complexity [8].

Finally, Poupard and Stern used lattices and rationals to build a key recovery system in [18].

1.2 Our Results

In this paper, we study a new version of a cryptocomputer. Instead of searching cryptosystems which only allow to perform $+$ and $\times$ operations over the integers, we cover additions and multiplications by a scalar of bounded rational numbers. We can thus construct an encrypted spreadsheet with bounded rationals and analyze its implementation. A limitation of our paper is to use *bounded* rationals. In fact, we need to recover the encoded rationals using a lattice and the output of the Gauss algorithm depends on the size of the shortest vector. However, thanks to the large bandwidth of Paillier scheme, a large number of operations can be done.

1.3 Notations and Definitions

Throughout this paper, we use the following notation: for any integer N,

- we use $\mathbb{Z}_N$ to denote the set of the integers modulo N,
- we use $\mathbb{Z}_N^*$ to denote the multiplicative group of invertible elements of $\mathbb{Z}_N$,
- we use $\varphi(N)$ to denote the Euler totient function, i.e. the cardinality of $\mathbb{Z}_N^*$,
- we use $\lambda(N)$ to denote Carmichael's lambda function defined as the largest order of the elements of $\mathbb{Z}_N^*$.

It is well known that if the prime factorization of an odd integer N is $\prod_{i=1}^{\eta} {q_i}^{f_i}$ then $\varphi(N) = \prod_{i=1}^{\eta} {q_i}^{f_i-1}(q_i - 1)$ and $\lambda(N) = \mathrm{lcm}_{i=1\ldots\eta}\left({q_i}^{f_i-1}(q_i-1)\right)$.

1.4 Outline of the Paper

In section 2 we recall preliminary tools such as the Paillier cryptosystem, homomorphic properties and the Gauss algorithm. Then in section 3, we show how to encode and decode rationals bounded by integers modulo N. Next, we describe additions and products in section 4. Finally, we describe practical parameters and our spreadsheet application.

2 Preliminary Tools

2.1 The Paillier Cryptosystem

Various cryptosystems based on randomized encryption schemes $\mathcal{E}(M)$ which encrypt a message M by raising a basis g to the power M have been proposed so far [13,4,7,23,15,16,17]. Their security is based on the intractability of computing discrete logarithm in basis g without a secret element, the secret key. Given this secret as a trapdoor, the computation becomes easy. We call those cryptosystems *trapdoor discrete logarithm schemes*. As an important consequence of this encryption technique, those schemes have homomorphic properties that can be informally stated as follows:

$$\mathcal{E}(M_1 + M_2) = \mathcal{E}(M_1) \cdot \mathcal{E}(M_2) \quad \text{and} \quad \mathcal{E}(k \cdot M) = \mathcal{E}(M)^k$$

Paillier has presented three closely related such cryptosystems in [17]. We only recall the first one. This cryptosystem is based on the properties of the Carmichael lambda function in $\mathbb{Z}_{N^2}^*$. We state its main two properties: for any $w \in \mathbb{Z}_{N^2}^*$,

$$w^{\lambda(N)} = 1 \bmod N, \quad \text{and} \quad w^{N\lambda(N)} = 1 \bmod N^2$$

Key Generation. Let N be an RSA modulus $N = pq$, where p and q are prime integers. Let g be an integer of order $N\alpha$ modulo N^2. The public key is $\mathsf{pk} = (N, g)$ and the secret key is $\mathsf{sk} = \lambda(N)$.

Encryption. To encrypt a message $M \in \mathbb{Z}_N$, randomly choose x in $\mathbb{Z}_N^*$ and compute the ciphertext $c = g^M x^N \bmod N^2$.

Decryption. To decrypt c, compute $M = \dfrac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \bmod N$ where the L-function takes as input an element from the set $\mathcal{S}_N = \{u < N^2 | u = 1 \bmod N\}$ and computes $L(u) = \frac{u-1}{N}$.

The integers $c^{\lambda(N)} \bmod N^2$ and $g^{\lambda(N)} \bmod N^2$ are equal to 1 when they are raised to the power N so they are N^{th} roots of unity. Furthermore, such roots are of the form $(1+N)^\beta = 1+\beta N \bmod N^2$. Consequently, the L-function allows to compute such values $\beta \bmod N$ and $L((g^M)^{\lambda(N)} \bmod N^2) = M \cdot L(g^{\lambda(N)} \bmod N^2) \bmod N$.

One can note that g can be set to $1 + N$ in order to make encryption more efficient since $\mathcal{E}(m, x) = (1 + mN)x^N \bmod N^2$.

Security. It is conjectured that the so-called composite residuosity class problem, that exactly consists in inverting the cryptosystem, is intractable. This problem is easier than inverting RSA with public exponent N modulo N, but it is unknown whether the two problems are equivalent. The semantic security is based on the difficulty to distinguish N^{th} residues modulo N^2. We refer to [17] for details.

2.2 The Gauss Algorithm

The Gauss algorithm solves the problem of finding a basis in a 2-dimensional lattice: it computes a basis achieving the two first minima. One can prove that the algorithm outputs such basis in polynomial time [14,22]. Vallée in [22] has proved that the number of iterations is in the worst case $3+\log_{1+\sqrt{2}} \max(\|\mathbf{u}\|, \|\mathbf{v}\|)$, and the number of iterations is constant on average [10]. Finally, let (u', v') be the first vector of the reduced basis. It is the shortest vector in the lattice; therefore $\gcd(u', v') = 1$.

```
Input: a basis (u, v) of a lattice L
Output: a reduced lattice basis (u, v)
if ||u|| < ||v||, then interchange u and v
do
   r ← u − qv where q = ⌊(u,v)/||v||²⌉
   u ← v
   v ← r
until ||u|| ≤ ||v||
return (u, v)
```

Where $\lfloor x \rceil$ represents the integer closest to the real x.

3 Encoding and Decoding Bounded Rationals by Integers Modulo N

In the Paillier cryptosystem presented above, the message M to be encrypted should be an integer modulo N. In this section we explain how a *bounded* rational t can be encrypted and recovered using the Paillier cryptosystem.

3.1 Encoding Rationals in Integers

Encoding of bounded rational. Let a bounded rational number $t = r/s$ where $r, s \in \mathbb{Z}$, $s \neq 0$, $-R < r < R$, $0 < s < S$ $\gcd(r, s) = 1$, and $\gcd(s, N) = 1$ since $s < p$ and $s < q$. We denote by t' the representation of t in $\mathbb{Z}_N$ which is defined as :

$$t' = rs^{-1} \bmod N \tag{1}$$

The integer s^{-1} exists since $\gcd(s, N) = 1$. By calculating $rs^{-1} \bmod N$, an integer $t' \bmod N$ is obtained which can be used for encrypting the rational t with the Paillier cryptosystem.

One can note that this encoding of rational respects the classical operations, namely if $\mathcal{E}(r)$ is a encrypted rational and $\mathcal{E}(i)$ an encrypted integer, then $\mathcal{E}(r) \times \mathcal{E}(i) = \mathcal{E}(r + i)$ and $\mathcal{E}(r)/\mathcal{E}(i) = \mathcal{E}(r - i)$.

Decoding of bounded rational. When retrieving the message information with the decrypting formula, the resulting message M equals $t' \bmod N$, *i.e.* $rs^{-1} \bmod N$.

Now the rational t must be recovered from $rs^{-1} \bmod N$. Recovery consists in fact in recovering $r < R$ and $s < S$ from $rs^{-1} \bmod N$. Recovery of r and s and thus of t can be correctly done by using two-dimensional lattice theory.

A lattice is a subgroup of some power $\mathbb{Z}$. It can be represented by a basis, two non-colinear vectors of the lattice.

Consider the two-dimensional lattice L defined as:

$$L : \forall (x, y) \in \mathbb{Z}^2, x = yt' \bmod N \tag{2}$$

The lattice L could then also be defined as:

$$L : \forall (x, y) \in \mathbb{Z}, sx = yr \bmod N \tag{3}$$

From formula 2 can be deduced that the vectors $(N, 0)$ and $(t', 1)$ form a basis of the two-dimensional lattice L.

From formula 3 can be deduced that the vector (r, s) also is a vector of the lattice L. Moreover, in order to obtain optimal values for r and s, the vector (r, s) should be a minimal vector of the lattice L. The shortest vector can be computed from the original basis and thus from t' and N by using the algorithm of Gauss [6]. The first vector of the basis corresponds to this rational. Therefore, we know that $\gcd(r, s) = 1$ and we obtain numerator and denominator that are coprime.

3.2 Decoding Rationals in Integers

In the above section we showed how lattice theory can be used to encode and recover rationals using the Paillier cryptosystem. In this section, we prove the unicity of the recovered solution (r, s) if the conditions of Theorem 1 are respected.

Theorem 1. *If $t' = rs^{-1} \bmod N$, $-R \leq r \leq R$ and $0 < s \leq S$, then the algorithm of Gauss uniquely recovers r and s provided $2RS < N$.*

Proof. We first prove that there cannot exist two linearly independent solutions in the lattice L generated by $(N, 0)$ and $(t', 1)$. Let (r_1, s_1) and (r_2, s_2) with $-R \leq r_i \leq R$ and $0 < s_i \leq S$ two different solutions. Such (r_1, s_1) and (r_2, s_2) would form a basis of a sub-lattice L' of the lattice L. The determinant N of the basis of L divides the determinant of the basis of L':

$$N \,|\det(L') \text{ where } \det(L') = \begin{vmatrix} r_1 & s_1 \\ r_2 & s_2 \end{vmatrix} = |r_1 s_2 - r_2 s_1|$$

Furthermore:

$$|r_1 s_2 - r_2 s_1| \leq 2RS < N$$

by assumption.

Therefore,

$$N\,|\det(L') < N \tag{4}$$

From formula 4, one can conclude that the determinant of the basis L' equals 0 and therefore the vectors (r_1, s_1) and (r_2, s_2) are colinear and cannot form a basis.

Thus, if the Gauss algorithm outputs (r, s) of the proper form, we are done. Otherwise, we have to argue in a more subtle manner and consider the lattice $\tilde{L}$ consisting of pairs (Sx, Ry) with $(x, y) \in L$. The determinant of this lattice is $\Delta = SRN$ and its shortest vector is of norm $\leq \sqrt{2}RS$ (coming from the encrypted rational). As before, it can be shown that there cannot be two linearly independent vectors of norm $\sqrt{2}RS$ since they would generate a sublattice with determinant $2R^2S^2 < \Delta$. Thus the Gauss algorithm run on $\tilde{L}$ outputs the correct rational number.

In order to recover the rational we need to restrict the rational to have numerator smaller than R and denominator smaller than S. Such numbers will be hereafter called *bounded rationals.*

4 Addition and Multiplication with Rationals

Applying Theorem 1, the implications for the homomorphic addition and product properties of the Paillier cryptosystem can be computed.

4.1 Addition

Lemma 1. *The algorithm of Gauss uniquely recovers the sum of ℓ terms of the form $r_i s_i^{-1}$ with $-R \leq r_i \leq R$ and $0 \leq s_i \leq S$ if $2(\ell+1)RS^{2\ell-1} < N$.*

Proof. Adding $r_1 s_1^{-1}$, $r_2 s_2^{-1}$, ..., and $r_\ell s_\ell^{-1}$ where $-R \leq r_i \leq R$ and $0 \leq s_i \leq S$ for $i = 1, \ldots, \ell$, results in:

$$\frac{r_1}{s_1} + \frac{r_2}{s_2} + \ldots + \frac{r_\ell}{s_\ell} = \frac{r_1 \prod_{i=2,\ldots,\ell} s_i + r_2 \prod_{i=1,\ldots,\ell, i\neq 2} s_i + \ldots + r_\ell \prod_{i=1,\ldots,\ell-1} s_i}{\prod_{i=1}^{\ell} s_i} \tag{5}$$

So with $|r_1 \prod_{i=2,\ldots,\ell} s_i + r_2 \prod_{i=1,\ldots,\ell, i\neq 2} s_i + \ldots + r_\ell \prod_{i=1,\ldots,\ell-1} s_i| \leq \ell RS^{\ell-1}$ and $|\prod_{i=1}^{\ell} s_i| \leq S^\ell$, theorem 1 implies that:

$$2\ell RS^{2\ell-1} < N \tag{6}$$

4.2 Multiplication

When we need to multiply a bounded rational $t = r/s$ by a known bounded rational $t' = r'/s'$, we transform t' into an integer in $\mathbb{Z}_N$ by computing $k = r' \times (s'^{-1} \bmod N) \bmod N$. Then, we raise $\mathcal{E}(r \times (s^{-1} \bmod N))$ to the power k.

Lemma 2. *The algorithm of Gauss uniquely recovers the product of two factors of the form $t_i = r_i s_i^{-1}$, r_i and s_i are integers and $-R < r_i < R$ and $0 < s_i < S$, if $2R^2S^2 < N$. If ℓ multiplications are done, the bound is $2R^\ell S^\ell < N$.*

Proof. Since the numerator is $r_1 r_2 < R^2$ and the denominator $s_1 s_2 < S^2$, we need to have $2R^2S^2 < N$. It is easy to show thanks to theorem 1, when we multiply ℓ bounded rationals, that

$$2(RS)^\ell < N \tag{7}$$

5 Choice of Parameters

Formula 6 and formula 7, respectively for the addition and product operations, show the limitations of the recovery method. Once the limit is reached, the homomorphic properties cannot be used anymore: a decryption and encryption step is needed before processing more operations.

For the addition can be concluded that the S parameter should be kept as small as possible in order to optimize the number of computations on ciphered data. For the product the numerator and denominator of the known rational should be kept as small as possible in order to optimize the number of computations on ciphered data. Indeed, before the numerator or denominator of the encrypted rational will be larger than the bound R or S, the spreadsheet must decrypt the rational, approximate it by a rational which have smaller numerator and denominator. We call this phase the "canonical form algorithm". One can use the continued fraction algorithm to approximate the decrypted rational by a bounded rational with smaller numerator and denominator. One can see that if we fix $|N| = 1024$, $R = 10^9 < 2^{30}$ and $S = 10^5 < 2^{17}$, one can only accept $\ell < \lfloor \frac{|N|-|R|-1}{2|S|} \rfloor + 1 \approx 30$ additions or $\ell < \lfloor \frac{|N|-1}{|R|+|S|} \rfloor \approx 21$ multiplications. This impacts the number of additions and multiplications that can be executed before the running of the canonical form algorithm.

A way to increase the number of possible additions or multiplications before runnig the canonical form algorithm is to use a cryptosystem with larger bandwidth. To this task, one can use a modulus with larger size or the technique of Damgård and Jurik [9] in order to increase the size of the bandwidth, N^{s-1}, using the same modulus N with computations done in N^s where s is some integer. This latter technique allows us to multiply the number of additions or multiplications by a factor $(s-1)$.

Finally, one can note that the bounds that we compute are not very tight. In practice, we can perform more additions or multiplications since the Gauss algorithm works nicer than the provable bounds.

6 Practical Implementation

The method presented in the above sections has been successfully implemented in a Microsoft Excel 2000 spreadsheet running under Microsoft Windows 2000. This spreadsheet illustrates the properties as explained in the paper throug computations which encrypted data.

Excel does not support computations with large numbers as required by the Paillier cryptosystem. In order to solve this problem, the spreadsheet is only a graphical interface showing the process and the computation results. This is realized by implementing some self-designed buttons in the spreadsheet to which Visual Basic commands are associated. Those Visual Basic commands perform three different steps:

1. Data from specific spreadsheet cells is put in some ASCII files;
2. One or several executables are started in the Microsoft Windows 2000 environment;
3. The results of the executables which is also stored in ASCII files is retrieved and shown in the spreadsheet.

The executables in step 2 are compiled with Borland C++ 5.0 for Windows functions. In order to make the necessary computations with large numbers, the NTL library [21] of C++ functions is used. One of the biggest problems in the spreadsheet is to prevent Excel from starting step 3 before step 2 has finished as those steps are independent processes for the Windows Operating System. In order to prevent this and make the spreadsheet run smoothly, some timers have been added in the Visual Basic code.

Although the chosen method is not the most efficient and even looks a little cumbersome, it makes both parts of the implementation (the graphical interface and the computations) independent not only from each other but also from the operating system, thus facilitating their reuse in future projects.

The spreadsheet consists of four actual sheets:

1. The addition homomorphic property of the Paillier cryptosystem with integers;
2. The product homomorphic property of the Paillier cryptosystem with integers;
3. The addition homomorphic property of the Paillier cryptosystem with rationals and their recovery;
4. The limitation of the recovery method for the additional homomorphic property of the Paillier cryptosystem.

The spreadsheet confirms the results from the paper and illustrates the practical possibilities of the presented method to use the Paillier cryptosystem in order to compute with encrypted data consisting of rational numbers.

7 Conclusion

We proposed a method to use the Paillier cryptosystem and its homomorphic properties with rational numbers. The recovery of the numbers make use of two-dimensional lattice theory and especially the Gauss algorithm. Furthermore, we showed the conditions which have to be checked in order to uniquely recover the rationals uniquely. We implemented the method in a practical spreadsheet which makes computations on ciphered rationals and illustrates the limitations.

An open problem is to find a method to round or truncate the encrypted computation results, resetting the size of the parameters on the way.

References

1. M. Abadi and J. Feigenbaum. Secure circuit evaluation : a protocol based on hiding information from an oracle. *Journal of Cryptology*, 2(1):1–12, 1990.
2. O. Baudron, P.A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System. In *PODC '01*. ACM, 2001.
3. O. Baudron and J. Stern. Non-interactive Private Auctions. In *Financial Crypto '01*, LNCS. Springer-Verlag, Berlin, 2001.
4. J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, 1987.
5. D. Boneh and R. Lipton. Searching for Elements in Black-Box Fields and Applications. In *Crypto '96*, LNCS 1109, pages 283–297. Springer-Verlag, 1996.
6. H. Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics 138. Springer-Verlag, 1993.
7. J. Cohen and M. Fisher. A robust and verifiable cryptographically secure election scheme. In *Symposium on Foundations of Computer Science*. IEEE, 1985.
8. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Eurocrypt '01*, LNCS 2045, pages 280–300. Springer-Verlag, 2001.
9. I. Damgård and M. Jurik. Efficient Protocols based on Probabilistic Encryption using Composite Degree Residue Classes. In *PKC '01*, LNCS 1992, pages 119–136. Springer-Verlag, 2001.
10. H. Daudé, P; Flajolet, and B. Vallée. An average-case analysis of the gaussian algorithm for lattice reduction. *Combin. Probab. Comput.*, 6(4):397–433, 1997.
11. J. Feigenbaum and M. Merritt. Open Questions, Talks Abstracts, and Summary of Discussions. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2:1–45, 1991.
12. P. A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Crypto '00*, LNCS. Springer-Verlag, 2000.
13. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28, 1984.
14. A. Joux. *La Réduction des Réseaux en Cryptographie*. PhD thesis, École polytechnique, 1993.
15. D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proc. of the 5th CCCS*, pages 59–66. ACM press, 1998.
16. T. Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998.
17. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, 1999.

18. G. Poupard and J. Stern. Fair Encryption of RSA Keys. In *Proceedings of Eurocrypt 2000*, Lecture Notes in Computer Science, pages 172–189. Springer-Verlag, 2000.
19. R. Rivest, L. Adleman, and M. L. Dertouzos. On Data Banks and Privacy Homomorphisms. In *Foundations of Secure Computation*, pages 169–179. Academic Press, 1978.
20. T. Sander, A. Young, and M. Yung. Non-Interactive CryptoComputing for NC^1. In *Proc. of the 31st STOC*. ACM, 1999.
21. V. Shoup. Number Theory Library (NTL). Can be obtained at `http://www.shoup.net`.
22. B. Vallée. Gauss' algorithm revisited. *J. Algorithms*, 12:556–572, 1991.
23. S. Vanstone and R. Zuccherato. Elliptic Curve Cryptosystem Using Curves of Smooth Order Over the Ring Z_n. *IEEE Transaction on Information Theory*, IT-43, 1997.

Privacy Tradeoffs: Myth or Reality?
(Panel Summary)

Rebecca N. Wright*[1], L. Jean Camp[2], Ian Goldberg[3], Ronald L. Rivest[4], and Graham Wood[5]

[1] Stevens Institute of Technology, Hoboken, NJ 07030
[2] Kennedy School of Government, Harvard University, L 213, 79 JFK Street, Cambridge, MA 02138
[3] Zero-Knowledge Systems, 888 de Maisonneuve East, 6th Floor, Montreal, Quebec, Canada H2L 4S8
[4] Massachusetts Institute of Technology, 545 Technology Square, Room 324, Cambridge MA 02139
[5] Appleby Spurling & Kempe, Cedar House, 41 Cedar Avenue, Hamilton, HM 12 Bermuda

Abstract. We discuss tradeoffs between privacy and other attributes such as security, usability, and advances in technology. We discuss whether such tradeoffs are inherent, or if it is possible to "have it all."

"You have zero privacy anyway. Get over it."
— Scott McNealy, 1999
Chief Executive Officer
Sun Microsystems

Changes in technology are causing an erosion of privacy. Historically, people lived in smaller communities and there was little movement of people from one community to another. People had very little privacy, but social mechanisms helped prevent abuse of information. As transportation and communications technologies developed, people began to live in larger cities and to have increased movement between communities. Many of the social mechanisms of smaller communities were lost, but privacy was gained through anonymity and scale.

Now, advances in computing and communications technology are reducing privacy by making it possible for people and organizations to store and process personal information, but social mechanisms to prevent the misuse of such information have not been replaced. While a major issue in computing and communications technology used to be how to make information public, we now have to work hard to keep it private. The main causes for this are the reduced cost of data storage and the increased ability to process large amounts of data. Nonetheless, one should not entirely abandon the hope for privacy. Rather, new ways of thinking are needed to find solutions based on a combination of technology, policy, and education to try to maintain or increase privacy and to provide

* (moderator)

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 147–151, 2003.

new social mechanisms. The problem is not the new technology itself, but rather using old models and old modes of thought in dealing with situations arising from new technology.

Privacy may mean different things to different people. Three different aspects of privacy are:

- **seclusion:** the desire to be left alone
- **property:** the desire to be paid for one's data
- **autonomy:** the ability to act freely

For individuals whose primary concern is seclusion, protection of their property will not suffice. For example, an individual who cares about seclusion will consider the receipt of any e-mail spam as a critical privacy violation, while one who cares about property may be satisfied to receive payment or discounts in exchange for receiving spam. Autonomy is an issue if people find their behavior is constrained by their concerns that their behavior is being tracked. A general definition that can capture most aspects of privacy is "the ability to control the dissemination and use of one's personal information."

We investigate the question of whether there are tradeoffs between privacy and other attributes, and if such tradeoffs are inherent. For example, there may be tradeoffs between security and privacy, as is frequently mentioned in the United States since the terrorist events of September 11, 2001. In order to protect national security, the argument goes, it is necessary to routinely perform authentication and identification of individuals, and to monitor their whereabouts and behavior, despite the privacy violation this creates. There are also potential tradeoffs between privacy and usability (as introducing privacy features may make systems more difficult to use), privacy and marketability (as customers may not be willing to pay extra for privacy-protecting solutions, and businesses may not be willing to give up collecting personal information they deem valuable), and even between different notions of privacy such as property vs. autonomy.

In order to hope to achieve privacy of data, several kinds of protection are needed. It is necessary to protect stored data, data in transit, and to have some control over the release of data. Protection of stored data and data in transit are well-studied and somewhat well-solved problems in the areas of computer security and network security, and are usually considered outside the scope of "privacy." Solutions usually involve the use of encryption and authentication to ensure that data is only sent to authorized parties and is only readable by those it is sent to. In contrast, most current privacy-oriented work, such as P3P and related tools [1], assumes data in transit and storage will be protected, and focuses on helping users to state their willingness to release data based on how that data will be used. However, there have been a number of cases where personal information was leaked in violation of a stated privacy policy due to a failure of computer security. In this sense, privacy requires security, and security violations can lead to privacy violations.

Current privacy-oriented solutions such as P3P deal primarily with the case of interaction between the *stakeholder*, whose personal data is involved, and

an enterprise such as a business whose Web site is being visited. We call such data "transaction data." A second class of data is "authored data," which is created by the stakeholder(s). In this case, property is usually the relevant privacy issue. Digital rights management, which is geared toward guaranteeing the stakeholders' payments, aims to provide solutions. Both with transaction data and authored data, a common theme in many protection approaches is to label data in some way with the identities of the stakeholders and a description of the policies regarding use of the information.

A newer—and more difficult to control—class of data is "sensor data," or data that is collected by some kind of sensor. Some examples include:

- video surveillance cameras: the use of video surveillance cameras has become more and more pervasive, and such cameras have become smaller and more easily hidden or overlooked [3]. The privacy impact of surveillance cameras can be even greater if used in conjunction with face-recognition technology.
- various data mining applications related to national security or marketing, in which data is collected from diverse sources and combined in such a way as to reveal information about individuals' preferences, habits, or activities.
- desktop or keystroke monitoring software: such software is often used for intrusion or misbehavior detection in the workplace. Often workers may not be aware that they are being monitored in this way.
- GPS transmitters: for example, on taxicabs in order to help dispatchers provide better service. Unless protected properly by encryption, information from such transmitters can also be used, for example, for outside observers to determine the destinations of particular passengers.
- Radio-frequency identification (RFID) tags: it seems likely that in the near future, most products manufactured will contain an inexpensive RFID tag that broadcasts a unique 96-bit serial number when queried. Given that such products might include the clothing we wear and the money we carry and exchange, queries to RFID tags could potentially be used to track individuals' locations and interactions.
- wireless PDA's and other devices that broadcast recognizable identification information: for example, such services might allow a user to receive information from local businesses as she walks down the street. There is a clear tradeoff here between a user being open to services from entities they have no prior trust relationship with, and the potential for privacy invasions. Current implementations and standards tend to favor service provision over privacy.
- iris scans: it may be possible to do iris scans of individuals at reasonably large distances without their cooperation or knowledge. Such systems could be used, for example, to determine whether an individual has previously entered a building, even without necessarily identifying the individual.

Sensor data presents a real and growing privacy threat. In sensor data, the identities of the stakeholders are not necessarily clear at time of creation, nor are the identities of the data collectors or even the existence of the sensors necessarily known to the stakeholders. Hence, any privacy approach that labels data with

the stakeholders at the time of creation in order to constrain later behavior appropriately cannot work. As the above examples make clear, this class of data is growing rapidly.

The privacy impact of huge amounts of sensor data could be tremendous, as sensor data often crosses the boundary between the "real world" and "cyberspace". We note that when sufficiently aggregated (particularly across multiple entities), transaction data can have many of the same properties as sensor data. It is for this reason that most privacy policies focus on when and how data will be shared with other parties. Relatedly, this is why people objected to systems such as DoubleClick's, that would track Web-site browsing history, even if only the IP address, rather than an individual's name, was associated with the transaction at the time of collection [2].

In many cases, product decisions by large companies or public organizations become de facto policy decisions as their products are widely adopted. Often such decisions are made without conscious thought or public discussion about the privacy impacts. This is particularly true in the United States where there is not a lot of relevant legislation regarding what is legal from a privacy perspective. As an example of this, consider the difference between the magnetic stripe cards used to pay for the Metro in Washington, DC and those in New York City. In the former case, card usage data is (reportedly) not stored on a per card basis, while in the latter, it is. These decisions were most likely made not on the basis of privacy, but rather because at the time of implementation in Washington, data storage and processing techniques were less advanced, while at the time of implementation in New York City, it was clear that data storage was possible and the processing might yield information that would help the transit system to run more efficiently. We note that the lack of privacy in New York City has been used with good outcome (for example, corroborating alibis of innocent suspects in criminal cases), as well as having the potential for bad uses. The point we wish to emphasize is not that one technology is obviously better than the other, but that important privacy-relevant decisions were made without public debate and awareness.

The main tradeoff for privacy is advancing technology which makes it possible to store and process large amounts of data. Even if it were possible to stop such technological advances, most people would not advocate this approach in order to protect privacy. Similarly, there are instances where health or security concerns can override privacy concerns. For example, if an unconscious person is admitted to a hospital emergency room, it is generally more important to allow the medical personnel access to the person's medical history than to maintain privacy of that information at all costs. Similarly, if there is a specific immediate terrorist threat, security concerns can temporarily override privacy concerns. Still, it is important not to blindly give up privacy in the name of security or other goals. For example, the privacy-invasive question "who is this person?" is not the same as the security-relevant question "is this a dangerous person?" Answering the former question instead of the latter both unnecessarily violates privacy and may miss some security-relevant threats, particularly in the case

of first offenses. Regarding the tradeoff between privacy and usability, a good approach to trying to achieve both is a layered one involving reasonable defaults, easy and extensive customizations, and possibly visualization tools to help the user understand privacy-relevant attributes.

In many cases, the tradeoffs are to cost or power rather than an inherent conflict with privacy. That is, the apparent tradeoff between security and privacy may really be two tradeoffs: one between security and money, and the other between privacy and money. Similarly, the tradeoff between privacy and usability may in fact be a conflict between the power of technology providers or governments to provide and support solutions that do not provide both privacy and usability, and users willingness or need to use them.

In summary, while there is sometimes an inherent tradeoff between privacy and other attributes, it is important to realize that often it is possible to achieve other goals in conjunction with privacy.

References

1. L. F. Cranor and R. Wenning, "Why P3P is a Good Privacy Tool for Businesses and Consumers", *Gigalaw.com*, `http://www.gigalaw.com/articles/2002/cranor-2002-04.html`, 2002.
2. P. Jacobus, "Gookies" targeted as Congress, advocates address Net privacy", *CNET News.com*, February 11, 2000.
3. A. Reeder, "To See and Be Seen", *The Atlantic Monthly*, Vol. 282, No. 1, p. 62, July 1998.

An Improved Fast Signature Scheme without Online Multiplication*

Takeshi Okamoto[1], Mitsuru Tada[2], and Atsuko Miyaji[1]

[1] School of Information Science,
Japan Advanced Institute of Science and Technology (JAIST),
{kenchan, miyaji}@jaist.ac.jp
[2] Institute of Media and Information Technology,
Chiba University,
mt@math.s.chiba-u.ac.jp

Abstract. In this paper, we propose a fast signature scheme which is derived from three-pass identification scheme. Our signature scheme would require a modular exponentiation as preprocessing. However, no multiplication is used in the actual (i.e. on-line) signature generation. This means that the phase involves only a hashing operation, addition and a modular reduction. So far, some fast signature schemes called *on the fly* signatures were proposed. In those schemes the modular reduction is eliminated in the on-line phase. Therefore, our approach to obtain the fast on-line signature is different from theirs. This paper is the first approach for the fast signature scheme without on-line modular multiplication.

Keywords: three-pass identification, digital signature, on-line modular multiplication, random oracle model and provable security

1 Introduction

Nowadays, a signature scheme is an important tool for secure communication. Consequently, we are longing for more compact signature schemes to spread public-key infrastructure (PKI) system. In this case, the compactness means the efficiency of both computational work and transmitted data size. Such a compactness gives users' convenience, and is acceptable for various application to capacity limited devices such as a smart card.

To consider the computational efficiency in signature schemes, let us focus on the signer's computational work in *generic digital signature schemes*[1]. In such a signature scheme, there exist two kinds of computation for the signer: it consists

* This work has been supported by the Telecommunications Advancement Organization of Japan under the grant for international joint research related to information-communications.

[1] As well as in [17], in this paper, *a generic (digital) signature scheme* means a signature scheme which can be derived from a three-pass identification scheme by using an appropriate hash function.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 152–167, 2003.

of *pre-computation* and *(actual) signature generation*. The pre-computation can compute values without a message to be signed, and execute during idle time and completely independent of message to be signed. This means that such a computational cost does not influence the real-time computing. We say that such a computation is the *off-line* processing. On the other hand, the (actual) signature generation does directly influence the processing time because a message is indispensable to compute. We say that such a computation is *on-line* processing.

To estimate the efficiency of a signature scheme, we should separately consider the two types of computation. Needless to say, the *fast on-line signature*[2] can make digital signatures much more practical in a variety of scenarios.

To realize the fast on-line signature generation, Girault [6] modified Schnorr's signature scheme [22] in which an RSA-modulus[3] is used instead of a prime modulus. This modification leads to no modulo reduction in the on-line signature generation. Therefore, in Girault's scheme, faster processing of the signature generation is possible than in Schnorr's one. In 1998, Poupard and Stern [19] investigated and gave provable security for Girault's scheme, and named that scheme GPS-scheme. Due to the description of [19], in this paper, we call a generic signature scheme in which modulo reduction is not necessary in the on-line signature generation, *on the fly* signature scheme.

In 1999, Poupard and Stern [20] proposed another on the fly signature scheme (PS-scheme for short), whose security relies on the difficulty of integer factoring. In 2001, Okamoto, Tada and Miyaji [15] proposed on th fly signature (OTM-scheme for short) by improving PS-scheme in the computational work and transmitted data size.

In this paper, we propose a fast signature scheme, which is derived from a three pass identification scheme. Our idea to reduce the on-line computation, completely differ from that of on the fly signatures. That is, the on-line multiplication is eliminated in our scheme, whereas the modular reduction is done in on the fly signatures. Up to the present, some fast signature schemes which have the property of on the fly signature, have been proposed. However, this paper is the first approach for the fast signature scheme without on-line modular multiplication.

To sum up, in the on-line computation, the modular reduction is used in our schemes, and multiplication is used in on the fly schemes. Now let us compare the computational efficiency between the multiplication and the modular reduction. In the multiplication, a recursive algorithm due to [8] reduces the complexity of the multiplying. On the other hand, in the modular reduction, we can use the efficient technique such as [1,11].

Those methods are further advantageous than [8] because a *single modulus* can be used in our schemes and many reductions are performed by using such a modulus. That is, in our scheme the modulus s is fixed because s is a secret key.

[2] In this paper, a *fast on-line signature* means the signature scheme which has the on-line computational efficiency.

[3] In this paper, we call a modulus to be a product of two distinct primes an *RSA-modulus*.

This property lead to a good computational efficiency for modular reduction. On the other hand, such an efficiency does not exist for the multiplication.

Consequently, the on-line modular reduction in our schemes is faster than the on-line multiplication in on the fly signatures from implementation point of view. For more detailed treatment of the above methods, see [9].

As for the security, our scheme is based on the integer factoring problem for RSA modulus n and provably secure in the random oracle model. To satisfy the security, our scheme uses a public key g with specific structure, called *asymmetric basis*, and which is a variant of [16]. This property leads to the good efficiency in terms of both size of data and amount of work.

Concrete to say, compared with OTM-scheme, the size of a signature in our signature can be reduced by at least 21%, and computational cost in our scheme for verification can be reduced by 32%, respectively. In the same way, compared with PS-scheme, the size of a secret key and a signature can be reduced by at least 68% and 58%, respectively. Furthermore, the computational cost for pre-computation and verification can be reduced by at least 83% and 78%, respectively.

This paper is organized as follows. In Section 2, we will review on the fly schemes. In Section 3, we will introduce our proposed identification scheme, and will give provable security for ours. In Section 4, we will introduce our signature scheme derived from our identification, and will discuss the security consideration. In Section 5, we will suggest practical values of our identification and the resulting signature scheme and will describe the implementation notes. In Section 6, we will introduce an efficient signature scheme which is optimized in terms of data size, and will evaluate the performance of our signature scheme by comparing with existing on the fly signatures: OTM-scheme, PS-scheme and GPS-scheme. The conclusion will be given in Section 7.

2 Previous Work

In this section, we briefly survey the previous works.

We first introduce some notations. The symbol $\varphi(\cdot)$ denotes Euler totient function, that is, $\varphi(n)$ is the number of the natural numbers less than n and coprime to n. The symbol $\lambda(\cdot)$ denotes so-called Carmichael function, that is, $\lambda(n)$ is the greatest number among the possible orders of elements in $\mathbb{Z}_n^*$. The order of an element $g \in \mathbb{Z}_n^*$ is represented as $\mathrm{Ord}_n(g)$. We denote by $|x|$, the binary length for a positive integer x. We say that p is a strong prime if $p = 2p'+1$ and both p and p' are primes.

Now let us recall the on the fly schemes (identification version).

GPS-scheme [19]: The public key is (n, g, v), where n is an RSA modulus, $g \in \mathbb{Z}_n^*$ is an element with high order, and $v = g^{-s} \bmod n$, where $s \in \mathbb{Z}_{2^k}$ is picked up at random. The secret key is s.

In identification step, the commitment is $x = g^r \bmod n$, where $r \in \mathbb{Z}_A$ is picked up at random, the random challenge is $e \in_R \mathbb{Z}_B$, and the answer is

$y = r + se$ (in $\mathbb{Z}$). A verifier finally checks whether $x = g^y v^e \bmod n$ holds or not. Here each parameter (A, B, k) satisfies $B \ll 2^k \ll A$.

It is known that if one sets up the parameters in GPS-scheme such that the scheme is as secure as the discrete logarithm problem for modulus n under the one key attack scenario [20], the scheme leads to the inefficiency for both the amount of work and transmitted data size. For more details, we refer to [16].

PS-scheme [20]: The public key is (n, g), where n is a product of strong primes p and q, and $g \in \mathbb{Z}_n^*$ is an element such that $\mathrm{Ord}_n(g) \in \{\lambda(n), \lambda(n)/2\}$. The secret key is $s = n - \varphi(n)$ $(= p + q + 1)$, where $|s| = k$.

In identification step, the commitment is $x = g^r \bmod n$, where $r \in \mathbb{Z}_A$ is picked up at random, the random challenge is $e \in_R \mathbb{Z}_B$, and the answer is $y = r + se$ (in $\mathbb{Z}$). A verifier finally checks whether both $y < A$ and $x = g^{y-ne} \bmod n$ hold or not. Here each parameter (A, B, k) satisfies $B \ll 2^k \ll A$.

PS-scheme has some drawbacks: As you see, a secret key in PS-scheme is $s = n - \varphi(n)$ and the size of s is very large, i.e. about $|n|/2$. This feature lead to inefficiency in view of the amount of work and transmitted data size.

The following OTM-scheme solves the above drawback.

OTM-scheme [15]: The public key is (n, g, z), where $n = \prod_{i=1}^{t} p_i$ for an integer $t \geq 2$, $g \in \mathbb{Z}_n^*$ is an element such that $q = \mathrm{Ord}_n(g)$ and $z \in \mathbb{Z}_{2^c}$ is picked up at random. The secret key is $s = z \bmod q$, where $|s| = k$.

In identification step, the commitment is $x = g^r \bmod n$, where $r \in \mathbb{Z}_{2^a}$ is picked up at random, the random challenge is $e \in_R \mathbb{Z}_{2^b}$, and the answer is $y = r + se$ (in $\mathbb{Z}$). A verifier finally checks whether both $y < 2^{a+1}$ and $x = g^{y-ze} \bmod n$ hold or not. Here each parameter (a, b, c, k) satisfies $2^b \ll 2^k \ll 2^a \ll 2^{bc}$.

The discussion for the above on the fly signatures and our proposed scheme, is given in Section 6.2.

3 Identification Scheme

In this section, we introduce our identification scheme. The security consideration is also discussed.

3.1 Protocol

Our identification scheme uses the parameters k, s, a and b with $2^{k-1} \leq s < 2^k \ll 2^a \ll 2^b$, and also uses slightly generalized asymmetric basis [16], which is defined as follows.

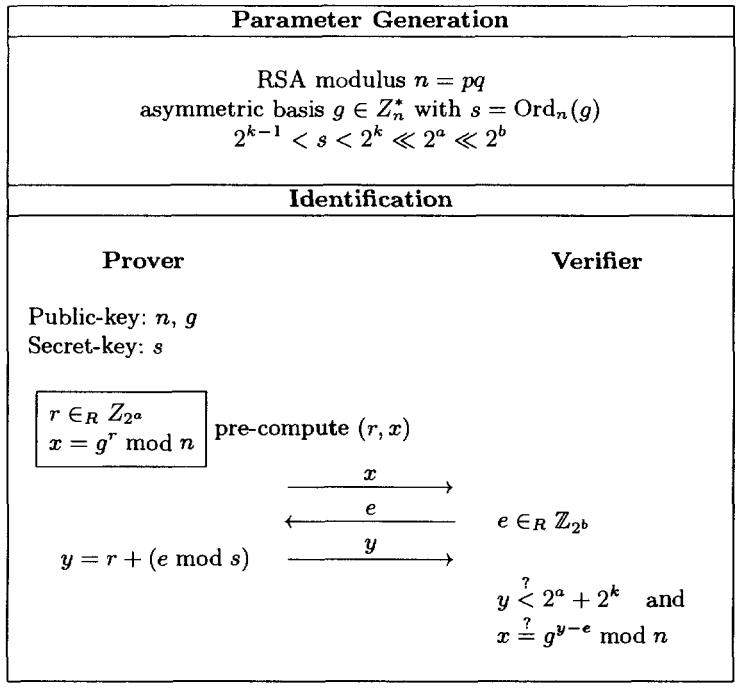


Fig. 1. Proposed identification scheme

Definition 1 (Asymmetric basis) Let n be an RSA modulus such that $n = pq$. Then we say that g is an asymmetric basis in $\mathbb{Z}_n^*$ if the multiplicity of 2 in $\mathrm{Ord}_p(g)$ is not equal to that of 2 in $\mathrm{Ord}_q(g)$. □

We now give our identification protocol.

Key generation step: The following steps are executed:

Step1 Pick up two same-size primes p and q, and compute $n = pq$.

Step2 Choose an element $g \in \mathbb{Z}_n^*$ which is an asymmetric basis in Z_n^*, where $s = \mathrm{Ord}_n(g)$.

Public-key/Secret-key: The public-key is (n, g) and the corresponding secret-key is s.

Identification step: A prover and a verifier execute the following steps:

Step1 The prover picks up a random number $r \in \mathbb{Z}_{2^a}$, computes the commitment $x = g^r \bmod n$ and sends x to the verifier.

Step2 The verifier picks up a random challenge $e \in \mathbb{Z}_{2^b}$ and sends e to the prover.

Step3 The prover computes $z = e \bmod s$ and an answer $y = r + z$ in $\mathbb{Z}$, and sends y to the verifier.

Step4 The verifier checks whether both $y < 2^a + 2^k$ and $x = g^{y-e} \bmod n$ hold or not. If both equations hold, the verifier accepts. Otherwise she rejects.

In Step3, the on-line multiplication for the prover is eliminated. This is the main idea of our scheme.

Note that, in the conventional identification schemes such as Schnorr [22] or Guillou-Quisquater [7], the challenge e can be a fixed constant. In this case, such schemes whould have some rounds. However, in our identification, e has the condition such that $2^k \ll e$. Therefore, the round of our identification is constant and fixed as one. We can say such a property indeed characterizes our schemes.

3.2 Security Analysis

We show that our proposed scheme is a three-pass honest-verifier statically zero-knowledge identification protocol. As a strategy, we would like to indicate that our scheme provides completeness, soundness and the statistical zero-knowledge property, respectively. To estimate the rigorous security, we follow the approach of Feige, Fiat and Shamir in [5].

We say that a positive function $f(k) : \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, if for any c, there exists a k_c such that $f(k) \leq k^{-c}$ for any $k \geq k_c$. Otherwise f is said to be *non-negligible*. $\|x\|$ denotes the absolute value of x.

Lemma 2 Let n be an RSA modulus and g be an asymmetric basis in $\mathbb{Z}_n^*$. Assume that we find $L > 0$ such that $g^L = 1 \bmod n$. Then we can construct a Turing machine M which on input n, g and L outputs a factor of n in time $O(|L||n|^2)$

Proof. This lemma is basically due to [16]. Hereafter, we describe how to construct M.

At first, M extract the odd part b of L, such that $L = 2^a b$. Since g is an asymmetric basis in $\mathbb{Z}_n^*$, it holds $g^{2b} = 1 \bmod p$ and $g^{2b} = 1 \bmod q$, and also holds $g^b = 1 \bmod p$ and $g^b = -1 \bmod q$. Then we have the following results: $p \mid g^b - 1$ and $n \nmid g^b - 1$. Consequently, M can find a factor of n by computing $\gcd(g^b - 1 \bmod n, n)$.

Note that the modular exponentiation algorithm and the extended Euclidean algorithm have a running time of $O(|L||n|^2)$ and $O(|n|^2)$, respectively. Hence M can execute the above steps in time $O(|L||n|^2)$. □

Theorem 3 (Completeness) In the proposed identification scheme, the prover who has a secret-key, and who follows the scheme, is always successfully accepted.

Proof. The answer y correctly computed by the prover, is $r + z$, where $z = e \bmod s$. Since $r < 2^a$ and $z < s < 2^k$, those conditions satisfy $y < 2^a + 2^k$. Furthermore, from the following equations:

$$g^{y-e} = g^{(r+z)-e} = g^{r+(e \bmod s)-e} = g^r = x \bmod n,$$

a correctly generated answer y always passes the verification. □

Theorem 4 (Soundness) If there exists a polynomial-time adversary $\tilde{P}$ which is accepted by honest verifiers with probability $> 1/2^b + \varepsilon, \varepsilon > 0$, where the average running time is T. Then, by using $\tilde{P}$, we can construct a polynomial-time machine M, which can figure out the factorization of public-key n, in expected time $O(T/\varepsilon + |n|^{O(1)})$.

Proof. Hereafter, we illustrate how to construct M. First M input (n, g) to $\tilde{P}$. Here we denote by ST, the current $\tilde{P}$'s state at this stage. *Ver* is a verification function, on input (x, e, y), outputs OK if those parameters are valid. Otherwise it outputs NG.

M executes the following algorithm and tries to obtain forged parameters.

```
% Algorithm 1
input result := NG, counter := 0
while (result = NG, counter < |n|/ε) do
pick up a random tape ω at random
x := P̃(ω)
pick up a random number e ∈ Z_{2^b} at random
y := P̃(e)
result := Ver(x, e, y)
counter := counter + 1
end while
output (ω, x, e, y)
end
```

If the forgery of Algorithm 1 is successful, M executes the next steps. Otherwise it stops (this time the attempt is failure). The probability with which M can obtain the forged parameters is estimated as:

$$1 - (1 - \varepsilon)^{\frac{|n|}{e}} \geq 1 - \left(\frac{1}{\tilde{e}}\right)^{|n|},$$

where $\tilde{e}$ is the natural logarithm.

Next, M initializes the $\tilde{P}$'s condition: set as ST. By using the same (ω, x) obtained by Algorithm 1, M tries to obtain forged parameters (x, e', y'). In this case, we denote by ε' the probability to obtain those parameters by one iteration.

M executes the following algorithm.

```
% Algorithm 2
input ω, result := NG, counter := 0
x := A(ω)
while (result = NG, counter < |n|/ε') do
pick up a random number e' ∈ Z_{2^b} at random
y' := P̃(e')
result := Ver(x, e', y')
counter := counter + 1
end while
output (e', y')
end
```

It can be formally proved that $\varepsilon' \geq \varepsilon/2$. For more details, see *Splitting Lemma* in [17]. In the same way as before, the probability with which M can obtain the forged parameters is $\geq 1 - \left(\frac{1}{e}\right)^{|n|}$.

As a consequence, the running time by executing Algorithm 1 and Algorithm 2 is $O(|n|t/\varepsilon)$ and the probability with which M can obtain (x, e, y) and (x, e', y') is estimated as $\geq \frac{1}{2}\left(1 - \left(\frac{1}{e}\right)^{|n|}\right)^2$.

Finally, M computes $L = \|(y - y') - (e - e')\|$. Here $L > 1$ is a multiple of $\mathrm{Ord}_n(g)$, that is, $g^L = 1 \bmod n$, and g is an asymmetric basis. Therefore, as in the consequence of Theorem 2, M can obtain the factorization of public in expected time $O(kT/\varepsilon + |n|^{O(1)})$. □

Theorem 5 (Zero-knowledge property) The proposed identification scheme provides honest-verifier statistical zero-knowledge property, if $2s/2^a$ is negligible.

Proof. We prove the above theorem along the line of Theorem 6 in [19]. To prove the theorem, we show that we can construct a polynomial-time machine (simulator) M which simulates the communication between the honest prover and a honest verifier.

In this case, M executes the following steps:

Step1 Pick up two random numbers: $e' \in \mathbb{Z}_{2^b}$ and $y' \in \mathbb{Z}_{2^a}$.
Step2 Compute $x' = g^{y'-e'} \bmod n$.
Step3 Output (x', e', y').

We denote by π, the probability in the communication between a honest prover and a honest verifier, that is, $\pi = \mathrm{Pr}\left[(x, e, y) = (\alpha, \beta, \gamma)\right]$. Then we have the following:

$$
\begin{aligned}
\pi &= \mathrm{Pr}\left[\begin{array}{c} \alpha = g^r \bmod n, \\ \beta = e, \\ \gamma = r + \Omega \end{array}\right] \\
&= \frac{1}{2^{a+b}} \cdot \sum_{\substack{0 \leq r < 2^a \\ 0 \leq e < 2^b}} \mathrm{Pr}\left[\begin{array}{c} \alpha = g^{\gamma - \Omega'} \bmod n, \\ \beta = e, \\ r = \gamma - \Omega' \end{array}\right] \\
&= \frac{1}{2^{a+b}} \cdot \mathrm{Pr}\left[\begin{array}{c} \alpha = g^{\gamma-\beta} \bmod n, \\ 0 \leq \beta < 2^b, \\ 0 \leq \gamma - \Omega' < 2^a \end{array}\right] \\
&= \frac{1}{2^{a+b}} \cdot \chi(\alpha = g^{\gamma-\beta} \bmod n) \\
&\quad \cdot \chi(0 \leq \beta < 2^b) \cdot \chi(\Omega' \leq \gamma < 2^a + \Omega'),
\end{aligned}
$$

where $(e \bmod q)$ and $(\beta \bmod q)$ are denoted by Ω and Ω', respectively. For a predicate Q, $\chi(Q)$ means the characteristic function of Q, that is, if Q is true, then $\chi(Q) = 1$, otherwise $\chi(Q) = 0$.

In the same way as above, the probability by M, that is, $\mathtt{Pr}\left[(x', e', y') = (\alpha, \beta, \gamma)\right]$ is denoted by π'. In this case, we have the following:

$$\begin{aligned}
\pi' &= \mathtt{Pr}\begin{bmatrix}\alpha = g^{y'-e'} \bmod n, \\ \beta = e', \\ \gamma = y'\end{bmatrix} \\
&= \frac{1}{2^{a+b}} \cdot \sum_{\substack{0 \le r < 2^a \\ 0 \le e < 2^b}} \mathtt{Pr}\begin{bmatrix}\alpha = g^{\gamma-\beta} \bmod n, \\ \beta = e', \\ \gamma = y'\end{bmatrix} \\
&= \frac{1}{2^{a+b}} \cdot \chi(\alpha = g^{\gamma-\beta} \bmod n) \\
&\qquad \cdot \chi(0 \le \beta < 2^b) \cdot \chi(0 \le \gamma < 2^a),
\end{aligned}$$

We whould like to estimate the distance between actual system and M. We define $\Delta = \sum_{\alpha,\ \beta,\ \gamma} \|\pi - \pi'\|$. From the results of π and π', we obtain

$$\Delta = \sum_{\substack{\alpha = g^{\gamma-\beta} \bmod n \\ 0 < \beta < 2^b \\ 0 \le \gamma < \Omega'}} \pi' + \sum_{\substack{\alpha \\ \beta \\ \Omega' \le \gamma < 2^a}} \|\pi - \pi'\| + \sum_{\substack{\alpha \\ \beta \\ 2^a \le \gamma < 2^a + \Omega'}} \pi.$$

Since

$$\left(\sum_{\substack{\alpha,\ \beta \\ 0 \le \gamma < \Omega'}} \pi'\right), \left(\sum_{\substack{\alpha,\ \beta \\ 2^a \le \gamma < 2^a + \Omega'}} \pi\right) \le \frac{\Omega'}{2^a} < \frac{q}{2^a}$$

and

$$\sum_{\substack{\alpha,\ \beta \\ \Omega' \le \gamma < 2^a}} \|\pi - \pi'\| = 0,$$

we can conclude $\Delta < 2q/2^a$. This proves the theorem. □

4 Signature Scheme

We introduce our signature scheme which is derived from the identification scheme proposed in Section 3.

4.1 Protocol

As well as conventional identification schemes such as Schnorr [22] or Guillou-Quisquater [7], our identification scheme in Section 3 can be turned into a signature scheme by using the technique in [4]. Then the challenge in the identification is replaced with an appropriate hash function. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^b$ be a hash function.

Key generation step: This is the same as our identification scheme.

Public-key/Secret-key: This is the same as our identification scheme.

Signature generation step: Suppose a signer which has a public-key and the corresponding secret-key, generates the signature of her message $m \in \{0,1\}^*$. Then she executes the following steps:

Step1 Pick up a random number $r \in \mathbb{Z}_{2^a}$ and compute $x = g^r \bmod n$.

Step2 Compute $e = \mathcal{H}(x, m)$.

Step3 Compute $z = e \bmod s$ and $y = r + z$ in $\mathbb{Z}$.

Signature: The signature for a message m is (x, e, y).

Verification step: Suppose a verifier which has the signer's public-key and the corresponding message, checks the validity of the signature for m. Then she executes the following steps:

Step1 Check whether $y < 2^a + 2^k$ holds or not. If the equation does not hold, then reject the signature and stop this protocol.

Step2 Compute $e' = \mathcal{H}(x, m)$ and $x' = g^{y-e} \bmod n$.

Step3 Check whether both $e = e'$ and $x = x'$ hold or not. If both equations hold, accept the signature. Otherwise reject it.

4.2 Security Analysis

In Section 3.2, we have proved that our identification scheme is three-pass honest-verifier statistically zero-knowledge identification protocol. This result includes all the properties to apply the technique of *the forking lemma* in [17]. Therefore, we can say that, in the random oracle model [2,17], our signature scheme is existentially unforgeable under the adaptive chosen message attack

Theorem 6 Let Q be the number of queries which a polynomial-time adversary $\mathcal{A}$ executing the adaptive chosen-message attack, can ask to the random oracle, and let R be the number of queries which $\mathcal{A}$ can ask to the actual signer. Assume that $2s/2^a$ is negligible. Also assume that, $\mathcal{A}$ can forge a signature with non-negligible probability $\varepsilon \geq 10(R+1)(R+q)/2^b$, where the average running time of $\mathcal{A}$ is T. Then we can construct a polynomial-time machine M which can factor n with non-negligible probability in expected time $O(QT/\varepsilon + |n|^{O(1)})$.

Proof. By the result of Theorem 5, the signature oracle in our signature scheme can be statistically simulated by a probabilistic polynomial-time machine M which works according to the protocol like [17].

When M uses $\mathcal{A}$, she can obtain two distinct signatures (x, e, y) and (x', e', y') such that $x = x'$, but $e \neq e'$. Then, we can get a multiple of $\mathrm{Ord}_n(g)$ such that $L > 1$ and $g^L = 1 \bmod n$. Here g is an asymmetric basis in $\mathbb{Z}_n^*$, therefore by the result of Lemma 2 we can find a factor of n in expected time $O(QT/\varepsilon + |n|^{O(1)})$. □

5 Parameter Generation

In this section, we describe the conditions of the parameters to keep the security in our identification and the corresponding signature scheme. We also show that how to implement the parameters.

5.1 Choice of Parameters

Parameters a and b: For the security reason, the values of a and b shall satisfy: $a = k + \kappa_1$ and $b = a + \kappa_2$, where k is a security parameter satisfying $|s| = k$ with $\mathrm{Ord}_n(g) = s$, and where both κ_1 and κ_2 are information leak parameters.

Parameters κ_1 and κ_2: By Theorem 5, we must set such that $1/2^{\kappa_1}$ is intractable. As for κ_2, let us first consider the following problem: given (n, g, α, κ_2), find β such that $g^\alpha = g^\beta \bmod n$, where $|\beta|$ is at least κ_2-bits smaller than $|\alpha|$. In our schemes, we must set κ_2 on the condition that the problem is hard to solve for the security parameter k. For implementation, we should take κ_1 and κ_2 greater than 64 and 24 bits, respectively.

Parameter s: Let us consider the attack which an adversary computes s only from the information of the public-key (n, g). We can see the algorithms to extract s, such as *Pollard lambda method* in [18] and *the baby-step giant-step method* in [10]. One may say that the former is better than the latter since it has same computational complexity (exponential-time: $O(\sqrt{s})$) but does not need large memory. The size of s shall be set up such that the above algorithms cannot apply for the security parameter k. For implementation, we should take $|s| = k$ greater than 160 bits for the security reason.

5.2 Implementation Notes

We first describe how to find p, q and an asymmetric basis g in $\mathbb{Z}_n^*$.

Step1 Pick up two primes $p = 2p'p'' + 1$ and $q = 2q'q'' + 1$ such that p' and q' are also primes, and p'' and q'' are odd numbers.

Step2 Choose $\alpha_p \in \mathbb{Z}_p^*$ satisfying $g_p = \alpha_p^{(p-1)/p'} \neq 1 \bmod p$. In the same way, choose $\alpha_q \in \mathbb{Z}_q^*$ satisfying $\alpha_q \neq q - 1 \bmod q$, $\alpha_q^{(q-1)/2} \neq 1 \bmod q$ and $g_q = \alpha_q^{(q-1)/2q'} \neq 1 \bmod q$.

Step3 Compute $n = pq$ and $g = q(q^{-1} \bmod p)g_p + p(p^{-1} \bmod q)g_q \bmod n$.

In Step3, g is computed by using the technique of Chinese Reminder Theorem (CRT). Note that $\mathrm{Ord}_p(g) = p'$ and $\mathrm{Ord}_q(g) = 2q'$. Therefore $\mathrm{Ord}_n(g) = \mathrm{lcm}(p', 2q') = 2p'q'$.

Next, we discuss secure hash algorithm which we should adopt. If $\mathcal{H}$ were an *ideal* hash function, then the proposed signature scheme would be *secure* under

the meaning of the description in Section 4.2. Since such a random function does not exist in the real world, in implementation, we are recommended to adopt MD5 by [21] or SHA-1 by [13], each of which is designed so that the algorithm can be a collision intractable hash function [3].

6 Efficiency for Signature Schemes

In this section, we first introduce the optimized signature scheme. We next evaluate the performance.

6.1 Optimized Scheme of Data Size

With respect to the signature scheme in Section 4, we can diminish the size of the signature. Consequently, communication lord is more efficient than before. We now focus on the following two parts:

Elimination of x: When we have two parameters e and y, the parameter x can be generated by computing $x = g^{y-e} \bmod n$. Therefore, the signature x is eliminated like conventional generic signature schemes such as Schnorr [22] or Guillou-Quisquater [7]. In this case, the signature for m consists of (e, y).

Using a short-size e: As for the signature scheme in Section 4, signature e is, in a certain sense, verbose. In our signature scheme, large-size e such as $2^a \ll e$ is actually needed in verification. Hence, we can use the following technique: we regard short-size e satisfying $2^k \ll e \ll 2^b$ as signature, and in verification, we extend the size of e by using appropriate hash function.

We use two hash functions $\mathcal{F} : \{0,1\}^* \to \{0,1\}^c$ and $\mathcal{G} : \{0,1\}^c \to \{0,1\}^b$, where c has the condition $2^c \ll 2^b$. The optimized signature scheme, which apply the above techniques, are as follows:

Key generation step: This is the same as our identification scheme.

Public-key/Secret-key: This is the same as our identification scheme.

Signature generation step: A singer executes the following steps:
Step1 Pick up a random number $r \in \mathbb{Z}_{2^a}$ and compute $x = g^r \bmod n$.
Step2 Compute $e = \mathcal{F}(x, m)$.
Step3 Compute $\epsilon = \mathcal{G}(e)$, $z = \epsilon \bmod s$ and $y = r + z$ in $\mathbb{Z}$.

Signature: The signature for a message m is (e, y).

Verification step: A verifier executes the following steps:
Step1 Check whether $y < 2^a + 2^k$ holds or not. If the equation does not hold, then reject the signature and stop this protocol.
Step2 Compute $\epsilon' = \mathcal{G}(e)$ and $e' = \mathcal{F}(g^{y-\epsilon'} \bmod n, m)$.

Step4 Check whether $e' = e$ holds or not. If the equation holds, accept the signature. Otherwise reject it.

As for the hash function $\mathcal{G}$, we should take c greater than 160 bits for implementation.

6.2 Performance

We evaluate the efficiency of our signature scheme by comparing existing on the fly signatures. Table 1 gives the performance of various schemes, such as OTM-scheme, PS-scheme and GPS-scheme, including ours.

The parameters in the schemes are set up as follows.

- Our scheme has $a = 224$ and $b = 248$ by taking $k = 160$, $\kappa_1 = 64$ and $\kappa_2 = 24$. Our scheme also use the technique of Section 6.1.
- OTM-scheme has $a = 104$, $b = 80$ and $c = 288$ by taking $k = 160$. To keep the same security as our scheme, in OTM scheme, n is a RSA modulus and g is an asymmetric basis in $\mathbb{Z}_n^*$. Furthermore, the size of public-key is optimized as follows. We regard actual public-key as (n, g), and z is computed by $z = \mathcal{H}'(n, g)$, where $\mathcal{H}'$ is a hash function $\mathcal{H}' : \{0,1\}^* \to \{0,1\}^c$.
- PS-scheme has $|A| = 656$ and $|B| = 80$ by taking $k = 513$.
- GPS-scheme has $|A| = 1184$ and $|B| = 80$ by taking $k = 1024$.

Additionally, we set $|n| = 1024$ for all schemes.

Next, we show the comparison between OTM-scheme (resp. PS-scheme and resp. GPS-scheme) and our signature.

OTM-scheme: One of the verifications in OTM-scheme is $x = g^{y-ze} \bmod n$, where in the index of g, we can see the multiplication of two parameters z and e. On the contrary, the verification in our scheme is $x = g^{y-e} \bmod n$, hence the multiplication in the index does not exist. The large-size index involved by the multiplication, lead to the inefficiency from both the amount of work and data size point of view. Nowadays, all the existing on the fly signatures have the same drawbacks.

Consequently, `CVF` and `SSig` in our scheme is superior to those in OTM-scheme. Since the public key in our scheme and that in OTM-scheme are (n, g) and (n, g, z), respectively, the number of the parameters in our scheme is smaller than that in OTM-scheme.

PS-scheme: The size of secret key in PS-scheme is only dependent on the modulus n, and that is considerably large (about $|n|/2$). This drawback leads to inefficient results with respect to the computational work (`CPC`, `CSG` and `CVF`) and the data size (`SSK` and `SSig`).

On the other hand, since PS-scheme is intended to be used with a modulus product of two strong primes, $g = 2$ is a correct basis and do not have to be included in the public key. Consequently, we can set `SPC` $= 1024$ for PS-scheme. Therefore, one may say that PS-scheme is more efficient than our scheme in terms of size of public key.

Table 1. Performance of signature schemes

Scheme	UMP	CPC ($\times M$)	CSG	CVF ($\times M$)	SPK (bits)	SSK (bits)	SSig (bits)
Our scheme	Integer factoring with g	61	248 mod 160	372	2048	160	304
OTM-scheme	Integer factoring with g	61	80 × 160	552	2048	160	384
PS-scheme	Integer factoring	381	80 × 512	1656	1024	513	736
GPS-scheme	Discrete log. for modulo n	381	80 × 1024	1796	3072	1024	1264

Abbreviation:

- UMP means the underlying mathematical problem that the signature scheme relies on for its security.
- CPC, CSG and CVF mean the computational cost for pre-computation, signature generation and verification, respectively
- SPK, SSK and SSig means the size of a public-key, a secret-key and a signature, respectively.
- M represents the computational cost for one multiplication under a 1024-bit modulus.
- α mod β represents the computational cost for modular reduction of an α-bit number and a β-bit number modulus.
- $\gamma \times \delta$ represents the computational cost for multiplication of an γ-bit number and a δ-bit number on $\mathbb{Z}$.

Notes:

- For all schemes in the table, we set up the parameter under the line of the one-key attack scenario [20].
- For respective computational cost, a primitive arithmetic of binary methods [9] are used, e.g. amount of work for g^{α} mod n is $\frac{3}{2}|\alpha|M$ if $|n| = 1024$. Of course there exist more sophisticated techniques which reduce the amount of computational work. However we think they estimate the concrete performance without loss of generosity.
- In UMP, integer factoring with g means that it is a variant of integer factoring problem on input RSA modulus n and the asymmetric basis g, outputs the factor of n.
- In CPC, the signer uses the technique of CRT. In this case, the signer must secretly have the factors of n, p and q.

GPS-scheme: Since the public key in GPS-scheme consists of three parameters such as (n, g, v), the size of the public key is SPK = 3072. Hence, GPS-scheme has the largest size for public-key of all the schemes in Table 1.

Note that, in the table, all the schemes are based on the one-key attack scenario. Consequently, GPS-scheme has provable security such that the scheme is as secure as the discrete problem for modulo n. However, the size of secret-key in GPS-scheme is considerably large: SSK = 1024. In the same reason as in PS-scheme, this result leads to the inefficiency mentioned above.

Table 1 show that our signature scheme is quite efficient from both the computational cost and the data size point of view.

We now give the concrete evaluation measured by comparing the integer factoring based scheme, OTM-scheme (resp. PS-scheme) and our scheme. Compared with OTM-scheme, our scheme enables the computational cost to be reduced by 32% for verification. For the data size, the signature size is reduced by 21%. In the following, compared with PS-scheme, our scheme enables the computational cost to be reduced by 83% for pre-computation and by 78% for verification. For the data size, the secret-key size is reduced by 68% and the signature size is 58%.

7 Conclusion

In this paper, we have proposed an efficient and fast signature scheme, which is derived from a three-pass identification scheme. Our proposed signature is, in a sense, a counterpart for on the fly signature schemes.

Compared between two types of signatures, the on-line computation in our scheme is smaller than that in on the fly schemes, because modular reduction is faster than multiplication from implementation point of view.

We have shown that our signature schemes are existentially unforgeable against any polynomial-time adversaries that can execute adaptive chosen message attack in the random oracle model. In this case, the underlying number theoretic problem is the integer factoring problem for an RSA modulus n.

We also have shown that our scheme is more efficient than on the fly signature schemes, in view of both the computational cost and the transmitted data size.

References

1. P. Barrett: "*Implementing of the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor*", Advances in cryptology - Crypto'86, Lecture Notes in Computer Science 263, Springer-Verlag, pp. 311–323, 1987.
2. M. Bellare and P. Rogaway: "*Random oracles are practical: a paradigm for designing efficient protocols*", Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS), 1993.
3. I. Damgård: "*Collision free hash functions and public key signature schemes*", Advances in cryptology - Eurocrypt'87, Lecture Notes in Computer Science 304, Springer-Verlag, pp. 203–216, 1988.

4. A. Fiat and A. Shamir: "*How to Prove Yourself: practical solutions of identification and signature problems*", Advances in cryptology - Crypto'86, Lecture Notes in Computer Science 263, Springer-Verlag, pp. 186–194, 1987.
5. U. Feige, A. Fiat and A. Shamir: "*Zero-knowledge proofs of identity*", Journal of cryptology, vol.1, pp. 77–95, 1988.
6. M. Girault: "*Self-certified public keys*", Advances in cryptology - Eurocrypt'91, Lecture Notes in Computer Science 547, Springer-Verlag, pp. 490–497, 1992.
7. L. C. Guillou and J. J. Quisquater: "*A 'paradoxal' identity-based signature scheme resulting from zero-knowledge*", Advances in cryptology - Crypto'88, Lecture Notes in Computer Science 403, Springer-Verlag, pp. 216–231, 1989.
8. A. Karatsuba and Yu. Ofman: "*Multiplication of multidigit numbers on automata*", Soviet Physics - Koklady, vol.7, pp. 595–596, 1963.
9. D. E. Knuth: "*Seminumerical Algorithms*", The art of computer programming, vol.2, Second edition, Addison-Wesley, 1981.
10. D. E. Knuth: "*Sorting and Searching*", The art of computer programming, vol.3, Second edition, Addison-Wesley, 1998.
11. P. Montgomery : "*Modular multiplication without trial division*", Mathematics of computation, vol.44, pp. 519–521, 1985.
12. D. Naccache, D. M'raihi, S. Vaudenay and D. Raphaeli : "*Can DSA be improved?*", Advances in cryptology - Eurocrypt'94, Lecture Notes in Computer Science 950, 1995.
13. National Institute of Standards and Technology (NIST): "*Secure hash standards (SHS)*", Federal Information Processing Standards, 1995.
14. K. Ohta and T. Okamoto: "*On Concrete Security Treatment of Signatures Derived from Identification*", Advances in cryptology - Crypto '98, Lecture Notes in Computer Science 1462, pp. 354–369, 1998.
15. T. Okamoto, M. Tada and A. Miyaji "*Proposal of Efficient Signature Schemes based on Factoring*", Trans. IPSJ, Vol.42 No. 8, pp. 2123–2133, 2001.
16. D. Pointcheval: "*The Composite Discrete Logarithm and Secure Authentication*", Advances in cryptology - PKC'00, Lecture Notes in Computer Science 1751, 2000.
17. D. Pointcheval and J. Stern: "*Security arguments for digital signatures and blind signatures*", Journal of cryptology, vol.13, no.3, Springer-Verlag, pp. 361–396, 2000.
18. J. Pollard: "*Monte Carlo methods for index computation mod p*", Mathematics of Computation, vol 32, pp. 918–924, 1978.
19. G. Poupard and J. Stern: "*Security analysis of a practical 'on the fly' authentication and signature generation*", Advances in cryptology - Eurocrypt'98, Lecture Notes in Computer Science 1403, Springer-Verlag, pp. 422–436, 1998.
20. G. Poupard and J. Stern: "*On the fly signatures based on factoring*", Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), pp. 48–57, 1999.
21. R. L. Rivest: "*The MD5 message-digest algorithm*", Internet Request for Comments, RFC 1321, 1992.
22. C. P. Schnorr: "*Efficient signature generation by smart cards*", Journal of cryptology, vol.4, Springer-Verlag, pp. 161–174, 1991.

Timed Release of Standard Digital Signatures

Extended Abstract

Juan A. Garay[1] and Markus Jakobsson[2]

[1] Bell Labs – Lucent Technologies, 600 Mountain Ave, Murray Hill, NJ 07974.
garay@research.bell-labs.com
http://www.bell-labs.com/user/garay
[2] RSA Labs, Bedford, MA 01730
http://www.markus-jakobsson.com

Abstract. In this paper we investigate the timed release of *standard* digital signatures, and demonstrate how to do it for RSA, Schnorr and DSA signatures. Such signatures, once released, cannot be distinguished from signatures of the same type obtained without a timed release, making it transparent to an observer of the end result. While previous work has allowed timed release of signatures, these have not been standard, but special-purpose signatures.
Building on the recent work by Boneh and Naor on timed commitments, we introduce the notion of a *reusable time-line*, which, besides allowing the release of standard signatures, lowers the session costs of existing timed applications.

1 Introduction

While probably the most important pillar of cryptography is the believed existence of hard problems, the notion of *moderately hard* problems also bears promise. A moderately hard problem is one which is not computationally infeasible to solve, but also not easy. Since Dwork and Naor introduced the notion for purposes of spam protection [DN92], considerable research has gone into the area, and the methods have been refined. A first type of work relating to timing mechanisms is that of proposing and developing suitable primitives, for which good lower bounds can be developed. Another avenue of research relates to applications that *use* timed mechanisms. This paper makes progress in both of these areas by extending the recent work of Boneh and Naor on timed commitments and timed signatures [BN00] to (1) allow for the reuse of their proposed timed structure, thus achieving lower session costs, and (2) allow for the robust timed release of standard signatures.

Background and motivation. We will follow the convention and refer to schemes with a moderately hard computational "trigger" as *timed*, since the effort involved relates closely to the time of performing the computation, and, functionally speaking, it is often the time rather than the effort that is relevant. To this

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 168–182, 2003.

end, primitives have been proposed that do not allow for easy parallelization of the problem solving. This shifts the emphasis from a *global* computational effort to the local effort, which is more closely related to actual time than to CPU time. For this purpose, problems based on modular exponentiation are believed to be good, as considerable effort has been invested in finding efficient exponentiation algorithms, and it is believed to be a problem not well suited for parallelization. For this reason, and following [RSW96,BN00], we base our work on iterated squaring.

While the computational lower bounds for the *problem solver* are paramount, and have been the focus of previous research, the computational costs of the *problem generator* (resp., the *solution verifier*) have not received the same attention. One focus of our paper is to curb the computational costs to generate and verify problems, while maintaining other desirable properties (such as protection against parallel attacks). Of these two costs, it is the problem generation cost that is the most difficult to reduce, and that which we concentrate on. We introduce the notion of *reusable time-lines*, which allow an amortization of the generation costs. With this benefit, we also get a reduced communication complexity, as large portions of the interaction and verification can be performed at setup, rather than at each session.

The resulting primitive can be used to reach a new goal: the timed release of *standard* digital signatures. While previous work has allowed timed release of signatures [Sha84,FS86,BN00], these have not been standard signatures, but special-purpose signatures. The robustness properties of our reusable time-lines allow us to get timed release of standard signatures; we show how to do it for RSA, DSS and Schnorr signatures. Such signatures, once released, cannot be distinguished from signatures of the same type obtained without a timed release. This makes the timed release transparent to an observer of the end result, and allows for an integration of timed release in legacy systems.

Our methods allow for easy integration with fair exchange methods, and they can also be used to obtain timed escrowing of ciphertexts (whether in conjunction with additional decryption keys or without). Such releases are efficient if certain encryption schemes (e.g., ElGamal) are employed. In this extended abstract we will concentrate on the signatures application.

Prior and related work. We already mentioned the work of Dwork and Naor [DN92] on moderately hard functions, which they use to combat junk mail.[1] The idea of "sending information into the future" is attributed to May [May93], who discussed several applications, mainly in the context of encrypting information.

Timed primitives have been designed for cryptographic key escrow. In [Sha95], Shamir suggested to escrow only partially a DES key, so that to recover the whole key, the government would need to first obtain the escrowed bits of the key, and then search exhaustively for the remaining bits. A problem with this type of approach is that the search can be parallelized, thus making it

[1] In this extended abstract, we only review time-related work that is more closely related to ours.

difficult to give a precise bound on the number of steps needed for the recovery. Bellare and Goldwasser [BG96,BG97] later suggested the notion of "time capsules" for key escrowing in order to deter widespread wiretapping, and where a major issue is the verification at escrow time that the right key will be recovered.

In another line of work, Rivest, Shamir and Wagner [RSW96] suggested "time-lock puzzles" for encrypting data, where the goal is to design puzzles that are "intrinsically sequential," and thus, putting computers to work together in parallel does not speed up finding the solution. They base their construction on a problem that seems to satisfy that: modular exponentiation. In their work, however, no measures are taken to verify that the puzzle can be unlocked in the desired time.

Using a function similar to that of [RSW96], Boneh and Naor [BN00] suggest the notion of (verifiable) *timed commitments*, an extension to the standard notion of commitments in which a potential forced opening phase permits the receiver to recover (with effort) the committed value vithout the help of the committer. They show how to use timed commitments to improve a variety of applications involving time, including timed signatures, contract signing, honesty-preserving auctions, and concurrent zero-knowledge. Their notion and construction of timed commitments is the starting point for our work.

Our work. More specifically, we build on the Boneh-Naor structure for timed commitments. We call this structure a "time-line." In a nutshell, a time-line is a vector of elements, where each element is obtained by iterated squaring of the previous element. The time-line is represented by its endpoints, and the commitment it corresponds to is a value on the line. This value can be computed by iterated squaring of one of the values representing the time-line.[2]

A first contribution of our paper is the notion of *derived* time-lines, which are time-lines that are obtained by "shifting" of a master time-line. The benefit of doing this is that while expensive proofs are needed to verify the correctness of time-lines in [BN00] for *each* commitment, we only have to perform this computation once (during a setup phase), after which a much simpler (and less expensive) verification of correct shifting can be used. Derived time-lines are "backwards compatible," in the sense that they can be readily used to solve the same problems addressed in [BN00].

Moreover, our construction allows for another application. The second – and main – contribution of our work is the use of our derived time-lines (and the corresponding commitments) for a robust release of signatures. While the signatures released in [BN00] are "special-purpose" signatures (designed with a timed release in mind), we can release *standard* signatures. Essentially, we use the committed value from the time-line as a "blinding" factor for the signature [Cha82]. However, providing the "connective tissue" between the time-line component and the signature component of the timed signature protocol is an important element of our construction. We demonstrate this by exhibiting a timed release

[2] In [BN00] this vector is used to perform timed commitments to arbitrary strings. Thus, what we call a (commitment to a) time-line is a simpler building block: given the time-line the committed value is fixed; we elaborate on this onwards.

of RSA, Schnorr and DSA signatures. For RSA signatures, we require for simplicity that the party releasing the signature (the committer) has generated the signature, while the release protocols for Schnorr and DSA signatures allow the release of signatures generated by third parties; we show how to do the third party release for RSA in the full version of the paper. Allowing the timed release of signatures by parties other than the signer may be a beneficial property in payment schemes, among other applications.

Organization of the paper. The cryptographic tools and assumptions that are used in our constructions are given in Section 2. The notion of a time-line and the protocol for new time-line generation and commitment are presented in Section 3. The definition of timed release of a standard signature, together with the protocols for timed RSA, Schnorr and DSA signatures, are given in Section 4.

2 Cryptographic Tools and Assumptions

Let N be a Blum integer, i.e., $N = pq$, where p and q are distinct primes each congruent to 3 mod 4. Recall the notion of a Blum-Blum-Shub (BBS) sequence $x_0, x_1, \cdots, x_n$, with $x_0 = g^2 \pmod N$ for a random $g \in \mathbb{Z}_N$, and $x_i = {x_{i-1}}^2 \pmod N$, $1 \leq i \leq n$. It is shown in [BBS86] that the sequence defined by taking the least significant bit of the elements above is polynomial-time unpredictable (unpredictable to the left and to the right), provided the quadratic residuosity assumption (QRA) holds.

The generalized BBS assumption. In [BN00], Boneh and Naor postulate the following generalization of unpredictability of BBS-type sequences. Let N and g be as above, and k an integer such that $l < k < u$. Then given the vector

$$< g^2, g^4, g^{16}, \ldots, g^{2^{2^i}}, \ldots, g^{2^{2^{k-1}}}, g^{2^{2^k}} > (\mathrm{mod} N), \tag{1}$$

the (l, u, δ, ϵ) generalized BBS assumption states that no PRAM algorithm whose running time is less than $\delta \cdot 2^k$ can distinguish the element $g^{2^{2^{k+1}}}$ from a random quadratic residue R^2, with probability larger than ϵ. The bound l precludes the parallelization of the computation, while the bound u precludes the feasibility of computing square roots through factoring. We refer to [BN00] for further details.

Equality of discrete logs. Our constructions will be using (statistical) proofs of knowledge (PK) of equality of two discrete logs modulo the same number (say, n), or in *different* moduli. Proofs for equality modulo the same number have been considered in [CEvdG87,CP92,Bao98]. We will use EQLOG-1(x, n) to refer to these proofs.

Proofs for equality of two (or more) discrete logs (alternatively, a discrete log and a committed number, or two committed numbers) in different moduli have been considered in [BT99,CM99]. Specifically, and following [BT99], let t, l and s be three security parameters, and let n_1 be a large composite number whose factorization is unknown to the prover, and n_2 be another large number, prime

or composite whose factorization is known or unknown to the prover. Let g_1 be an element of large order of $\mathbb{Z}_{n_1}^*$ and h_1 be an element of the group generated by g_1 such that both the discrete logarithm of g_1 in base h_1 and the discrete logarithm of h_1 in base g_1 are unknown to the prover. Let g_2 be an element of large order of $\mathbb{Z}_{n_2}^*$. Assume that the prover knows an integer $x \in \{0, \ldots, b\}$. It is shown in [BT99,CM99] how to give an efficient (non-interactive) statistical proof of knowlege

$$\mathrm{PK}(x, r_1 : y_1 = g_1^x h_1^{r_1} \bmod n_1 \wedge y_2 = g_2^x \bmod n_2),$$

where r_1 is randomly selected from $\{-2^s n_1 + 1, \ldots, 2^s n_1 - 1\}$, and where completeness holds with probability at least $1 - 2^{-l}$ and soundness holds with probability at least $1 - 2^{-(t-1)}$. We will use EQLOG-2(x, n_1, n_2) as a short form for the above proof. The role of n_1, whose factorization is unknown to the prover, is that of a "helper" modulus, and can be constructed as in [CM99]. In our constructions we will omit a reference to the helper modulus for simplicity.

Regarding the size of x, however, what is proven is that $x \in [-2^{t+l}b, 2^{t+l}b]$ – i.e., an $O(2^{t+l})$ *expansion rate.* In our applications, we will have the additional requirement that the discrete log lie in a specific bounded range, which leads us to the next building block.

Proof that a discrete log lies in an interval. The idea of checking whether a committed integer lies in a specific interval has many cryptographic applications. It was first developed in [BCDvdG87], and has been later studied in [CFT98, CM99,Mao98,Bou00]. In [Bou00], Boudot presents an efficient method to prove that a committed number $x \in I$ belongs to I, and not to a larger interval (i.e., an expansion rate of 1).

Specifically, let t, l and s be three security parameters, and let n_1 be a large composite number whose factorization is unknown to the prover, as above. Let integer $x \in [a, b]$ and $y_1 = {g_1}^x {h_1}^{r_1} \bmod n_1$, where r_1 is randomly selected from $\{-2^s n_1 + 1, \ldots, 2^s n_1 - 1\}$. It is shown in [Bou00] how to give an efficient (non-interactive) statistical proof of knowlege

$$\mathrm{PK}(x, r_1 : y_1 = {g_1}^x {h_1}^{r_1} \bmod n_1 \wedge x \in [a, b]).$$

Combined with the proof of knowledge of discrete logs in different moduli mentioned before, the above protocol can be used to prove that a discrete logarithm modulo p (a prime number or a composite number whose factorization is known to the prover) belongs to an interval. Refer to [Bou00] for details. We use INTVL$(x \in [a, b])$ as a short form for the above proof.

3 Time-Lines

In this section we define the notions of a *time-line* and the *commitment* to a time-line. In a nutshell, a time-line is the partial description of a BBS sequence, as given by the input vector (1) in the generalized BBS assumption. In [BN00]

this vector is used to perform *timed commitments*[3] to arbitrary strings. The simpler building block of a time-line commitment (defined below) will be useful for this paper's main result, namely, the timed release of standard signatures (Section 4); it will also allow us to make timed commitments (as well as the other applications in [BN00]) more efficient (Section 3.2).

Let N be a Blum integer and g an element of large odd order in $\mathbb{Z}_N^*$ (see [BN00] for ways for choosing g so as to guarantee this). A *value-hiding time-line* (*time-line* for short) is a partial description of a BBS sequence, given by the sub-sequence $\{g^{2^{2^i}}\}_{i=0}^{k} \pmod N$. We call the value $g^{2^{2^k-1}}$ the time-line's *hidden value*.[4] Informally, we say that a value *lies on* a time-line if it can be obtained by iterated squaring of the time-line's start value.

Now the notion of a (T, t, ϵ) *time-line commitment* naturally follows. It enables the committer to give the receiver a timed commitment to a hidden value. At a later time, she can reveal this value and prove that it's the correct one. However, in case the committer fails to reveal it, the receiver can spend time T and retrieve it.

As in the case of the timed commitment of [BN00], a (T, t, ϵ) time-line commitment consists of three phases: the **commit** phase, where the committer commits to the time-line's hidden value by giving the receiver the description of the time-line and a proof of its well-formedness; the **open** phase, where the committer reveals information to the receiver that allows him to compute the hidden value — and be convinced that this is indeed the committed value; and the **forced open** phase, which takes place when the committer refuses to execute the open phase; in this case, the receiver executes an algorithm that allows him to retrieve the hidden value, and produces a proof that this is indeed the value.

A time-line commitment scheme must satisfy the following security constraints:

Binding: The value committed to is uniquely defined by the commitment. In particular, it is not possible to open up the commitment to a value other than that which will be obtained by the forced opening of the commitment (corresponding to the iterated squaring of the time-line's starting value.)

Soundness: After receiving the time-line, the receiver is convinced that the forced open phase will produce the hidden value in time T.

Privacy: Every PRAM algorithm whose running time is at most $t < T$ on polynomially many processors, given the transcript of the time-line commit protocol as input, will succeed in computing the time-line's hidden value with probability at most ϵ.

Proving the well-formedness of a time-line, as described below, involves a computational effort. However, once a time-line is established and its properties verified (call it the *master* time-line), we can generate subsequent time-line

[3] A timed commitment is an extension to the standard notion of commitments in which a potential forced opening phase permits the receiver to recover (with effort) the committed value vithout the help of the committer [BN00].

[4] The notion easily extends to more (polynomially many) hidden values. For simplicity, we just consider one, as this will be enough for our main application.

commitments at a low cost. Intuitively, the idea is for the committer to create a new time-line from the master time-line by applying a secret transformation value – the *shifting* factor – to the end points of the master time-line, in such a way that verification of the new time-line's properties is easy. We now present the protocols for time-line commitment in more detail.

3.1 Our Time-Line Commitment Protocol

During the commitment setup phase below, the master time-line is established and its well-formedness proven. The setup process follows the commitment protocol of [BN00] almost *verbatim*, except for modifications that are needed for the subsequent, and derived, time-line commitments. (These modifications make it possible to use (and re-use) the time-lines for transparent release of standard signatures, as will be shown later.)

Setup. Let $T = 2^k$ be an integer, and N as defined above, i.e., $N = pq$ for p, q two random n-bit primes such that $p = q = 3 \bmod 4$. We assume N is a publicly known modulus, associated either with time-line commitments in general, or with time-line commitments from one particular user. The generator g of the master time-line is chosen so that the largest prime divisor of the order of g in $\mathbb{Z}_N^*$ is large. Specifically, in [BN00] the party performing the setup picks a random $f \in \mathbb{Z}_N$ and computes $g = f^{(\prod_{i=1}^{p} q_i^n)} \bmod N$, where $q_1, q_2, \cdots, q_p$ are all the primes less than some bound B. Upon receiving f and g the receiver is assured that the order of g in $\mathbb{Z}_N^*$ is not divisible by any primes smaller than B. Next, the following steps are performed by the party generating the master time-line.

1. *Compute master time-line.* Compute $M_i = g^{2^{2^i}} \bmod N$, $0 \leq i \leq k$, $m_{1\text{st}} = g^{2^{2^k-1}} \bmod N$, and $m_{2\text{nd}} = g^{2^{2^k-2}} \bmod N$.

   If the party knows $\varphi(N)$ (as in the case where a trusted party generates the master time-line), this computation can be performed by computing $a_i = 2^{2^i} \bmod \varphi(N)$ and then $M_i = g^{a_i} \bmod N$. If $\varphi(N)$ is unknown to the party, then it can be performed by iterated squaring.

2. *Prove well-formedness.* Generate a proof that $M_i = g^{2^{2^i}} \bmod N$, for $0 \leq i \leq k$. This is done by showing that each triple $< g, M_i, M_{i+1} >$, $0 \leq i < k-1$, is of the form $< g, g^x, g^{x^2} >$, for some x.[5]

3. *Output master time-line.* Output $\{M_i\}_{i=0}^{k}$, $m_{1\text{st}}$ and $m_{2\text{nd}}$, along with the above proof of well-formedness, which we assume is non-interactive. Before using the master time-line, any party checks that ${m_{2\text{nd}}}^2 = m_{1\text{st}} \ (\text{mod}N)$, ${m_{1\text{st}}}^2 = M_k \ (\text{mod}N)$, and that the proof is correct.

We now show how to generate a new time-line based on the master time-line.

[5] These k proofs are based on the classical zero-knowledge proof that a tuple is a Diffie-Hellman tuple, take fours rounds, and can be performed in parallel – see [BN00] for details. The requirement that the smallest prime divisor of g's order is large is needed for the soundness of these zero-knowledge proofs [CP92]; see also [GMP01].

Commit phase. The committer performs the following steps:

1. *Choose shifting factor.* The committer picks at random a value $\alpha \in \mathbb{Z}_{\tilde{\varphi}}$, where $\tilde{\varphi} = \varphi(N)$ is used if known, and otherwise $\tilde{\varphi} = \lfloor N - 2\sqrt{N} \rfloor$.
2. *Compute new time-line.* The committer computes $h = g^{\alpha}$, $R_{k-1} = (M_{k-1})^{\alpha}$, $r_{\text{aux}} = (m_{\text{2nd}})^{\alpha}$, $r = (m_{\text{1st}})^{\alpha}$, and $R_k = (M_k)^{\alpha} \pmod N$. He outputs h, R_{k-1} and R_k, and **keeps** r **and** r_{aux} **secret.** r is the new time-line's hidden value. (r_{aux} will be used in our timed signature application of Section 4.)
3. *Prove well-formedness of new time-line.* The committer proves to the receiver that
$$\log_g h = \log_{M_{k-1}} R_{k-1} = \log_{M_k} R_k \, (= \alpha),$$
i.e., that the new time-line is correctly derived from the master time-line. He uses EQLOG-1(α, N) for this proof.

Sometimes we will use TL_{α} to refer to the time-line generated from the master time-line by applying α, and $\text{TL}_{\alpha}(r)$ to refer to the commitment to the time-line's hidden value. We will drop the subscript when clear from the context.

Open phase. The committer sends α and $f' = f^{2^{(2^k-1)}}$ to the verifier, who computes $r = (f'^{\alpha})^{\prod_{i=1}^{p} q_i^{n_i}} \bmod N$, and checks that $h = g^{\alpha} \bmod N$ and $r^2 = R_k \bmod N$. At this point the receiver has a square root of R_k. Having odd order ensures that r is in the subgroup generated by h.

Forced open phase. The user computes r from R_{k-1} by repeated squaring mod N; specifically, $2^k - 2$ operations.

Theorem 1. *Assume the hardness of the discrete logarithm problem and that the (l, u, δ, ϵ) generalized BBS assumption holds. Then, for $k > l$, the scheme above is a secure (T, t, ϵ) time-line commitment scheme with $t = \delta \cdot 2^k$ and $T = M(u) \cdot 2^k$, where $M(u)$ is the time it takes to square modulo an u-bit number.*

Proof (sketch). Note that, by construction, the order of h does not have any small prime divisors. Thus, the EQLOG-1 proofs in step 3 of the commit phase guarantee that h, R_{k-1} and R_k lie on the new time-line. This gives soundness since it proves that the beginnings of the time-lines (g, resp., h) are related to each other like the ends of the time-lines ($g^{2^{2^k}}$, resp., R_k). Thus, if the master time-line is correct, so is the derived time-line, since it establishes that $R_k = h^{2^{2^k}}$.

For the binding property, during the open phase, the receiver receives f' which leads to r satisfying $r^2 = R_k \bmod N$. Furthermore, r has odd order in $\mathbb{Z}_N^*$. During the commit phase, the verifier is also assured that h has odd order in $\mathbb{Z}_N^*$, and that R_k is in the subgroup generated by h (soundness property). The claim then follows since R_k has a unique square root in the subgroup (odd order), and in $\mathbb{Z}_N^*$ there can be at most one square root of R_k of odd order.

For privacy, the generalized BBS assumption protects from computing r from R_{k-1} in a number of operations significantly less than 2^k squarings, and the

difficulty of factoring from computing it from R_k; the hardness of the discrete logarithm problem prevents any information about α to be leaked. ∎

3.2 Efficiency and Applications of Time-Line Commitments

In [BN00], Boneh and Naor show how time-lines can be used for a variety of applications involving time, including timed commitments, timed (non-standard) signatures, contract signing, collective coin flipping, honesty-preserving auctions, and concurrent zero-knowledge. Specifically, to perform a timed commitment to a message m of length ℓ (assume for simplicity that $\ell \leq |N|$), the committer hides the message by computing $c_i = m_i \oplus g^{2^{2^k - i}}$, $1 \leq i \leq \ell$, and sends c to the receiver together with the time-line parameters.

The timed signature scheme of [BN00] is obtained from the timed commitment scheme above as follows. To obtain a timed signature on a message m, the signer first performs a timed commitment to a random secret string m', thus obtaining c', and then signs the pair (m, c') – call this signature $\mathrm{Sig}(m, c')$. Upon retrieving m' from the commitment's open (or forced open) phase, the verifier submits the triple $< m', c', \mathrm{Sig}(m, c') >$ as the proof of a valid signature on m.

Note that our time-line commitments (alternatively, derived time-lines) described above can be used in the same way to obtain timed commitments to arbitrary strings (resp., timed signatures), at the expense of an additional hardness assumption. The efficiency gains, however, will be substantial, as we would essentially perform the large portion of the work only once, in the setup session, and then amortize the cost of this over many re-uses of the master time-line. More specifically, to prove the well-formedness of a time-line of "length" k (i.e., requiring $T = 2^k$ work) the original Boneh-Naor scheme requires to repeat many times a protocol with k proofs of equality of discrete logs. This is the same cost that our time-line commitment setup phase incurs (step 2 of setup phase). However, once the setup is performed and the master time-line established, we only perform *two* such equality of discrete log proofs for each time-line re-use (specifically, to prove the consistent application of the shifting factor α). The open phases, whether forced or standard, have very similar complexity in the two schemes.

In the next section, we give another important application of time-line commitments: how to generate timed signatures in a "transparent" way, i.e., with no modifications to the arguments or definition of a valid signature.

4 Timed Release of Standard Signatures

Our approach to allow for the timed release of standard signatures (or other cryptographic function) is to use the hidden value from the time-line commitment, r, as a *blinding* factor [Cha82] for the signature. However, in order to obtain robustness for this operation, one has to prove that r is uniquely defined, that is, obtainable from both forced and standard decommitment, and correctly

applied to blind the signature (or other function). Secondly, the prover has to perform a signature-dependent proof that r is the blinding value used to derive the blinded signature given to the receiver, and thus, that the receiver will be able to obtain the unblinded signature once he has computed or obtained the blinding factor r. Since operations will be taking place in different moduli, care has to be taken here to rule out the possibility that a value congruent to r, but not equal to r, is used for the blinding. Before showing how these objectives are achieved, we give a more formal definition of our timed signature scheme.

Given a regular signature scheme that allows for blinding (e.g., RSA, Schnorr, DSA), let $\mathrm{Sig}(m)$ denote the signature on message m, generated with the signer's private key, and which verifies with the signer's public key. Our (T, t, ϵ) timed signature scheme will consist of the following phases:

Commit phase: The signer performs a (T, t, ϵ) time-line commitment, and uses the time-line's hidden value r to blind the signature — call it $\widetilde{\mathrm{Sig}}_r(m)$. The signer gives the pair $< \mathrm{TL}(r), \widetilde{\mathrm{Sig}}_r(m) >$ to the verifier, together with the proofs of well-formedness of the time-line, uniqueness of the blinding factor, and correctness of the blinding operation.

Open phase: The signer reveals r by opening the time-line commitment. The verifier uses r to unblind the signature and obtain $\mathrm{Sig}(m)$.

Forced open: The verifier uses the forced open algorithm from the time-line commitment to retrieve r, and uses it to unblind the signature as above.

Note on blinding values. In the following, we describe the blinding and recovery of RSA, Schnorr and DSA signatures. For all of these, we note that the blinding factor r that the verifier obtains from the time-line commitment is applied "as is," i.e., without any modification to make it an exponent of the right length; in particular, the exponent may be much larger than the order of the group. Similarly, in the preparation of the blinded signature, the blinding and all proofs of correct blinding are performed with respect to this unreduced value. While the prover may in some instances use a number not equal to r, but merely congruent, the structure of our proofs guarantee that the blinding and unblinding portions cancel out. Thus, the resulting unblinded signature is the expected and valid signature committed to. The fact that r is potentially much larger than a normal blinding factor for the particular signature scheme only amounts to some extra work in the blinding and the unblinding, and a skewing of the probability distribution of blinding factors [6]

4.1 Timed RSA Signatures

Before we recall the RSA parameters, we introduce some additional elements that will be used in the proof of uniqueness of the blinding factor. (These will be

[6] Skews in the probability distributions of exponents have recently been shown to be problematic, but only in situations with a very large reuse of the same exponent (such as repetitive signing) [Ble00]. We note that each blinding factor is used only once, and on signatures whose distribution is (assumed to be) correct, and not affected by the blinding. Therefore, the skewed distribution does not represent a security concern here.

common to the other signature schemes.) Let N be the publicly known modulus associated with time-line commitments from Section 3.1. Let $N' = N^2$ be a *second*, auxiliary modulus, and $g\prime = N + 1$, as in [Pai99]; recall that the order of the subgroup generated by $g\prime$ is equal to N. We will be using $g\prime$ and N' to perform auxiliary (standard) commitments, which in turn will allow us to establish the desired relation between the time-line's hidden value and the blinding factor.

Now let n be an RSA modulus, (e, n) be the signer's public key, and d his secret key, chosen in such that way that $m^{ed} = m \bmod n$ for all values $m \in \mathbb{Z}_n$. The signer's signature on a (hashed) message m is therefore $s = m^d \bmod n$.

Commit phase. We assume for simplicity that the signer and the verifier have already done a time-line commitment TL(r). The signer performs the following steps:

1. *Normal signature generation.* Let m be the message to be signed. The signer computes $s = m^d \bmod n$.
2. *Application of blinding factor.* The signer blinds the signature by computing $\tilde{s} = s^{1/r} \bmod n$, where r is the time-line's hidden value, and sends the pair $(m, \tilde{s})$ to the verifier.
3. *Auxiliary commitments and proof of uniqueness.* The signer computes

$$b_{\text{aux}} = g\prime^{r_{\text{aux}}} \bmod N',$$
$$b = g\prime^{r} \bmod N', \text{ and}$$
$$B = g\prime^{R_k} \bmod N'.$$

   The signer then proves that $\log_{g\prime} b_{\text{aux}} = \log_{b_{\text{aux}}} b \ (= r_{\text{aux}})$, using EQLOG-1-$(r_{\text{aux}}, N')$; that $\log_{g\prime} b = \log_b B \ (= r)$, using EQLOG-1$(r, N')$; and that r_{aux} lies in the right interval using INTVL$(r_{\text{aux}} \in [0, N-1])$.[7]
4. *Proof of correct blinding.* The verifier computes $X = \tilde{s}^e$. The signer proves that $\log_X m = \log_{g\prime} b \ (= r)$, using EQLOG-2$(r, n, N')$, and that r lies in the right interval using INTVL$(r \in [0, N-1])$.

Open and forced open phases. The verifier obtains r from the signer or from the time-line forced open algorithm. He then unblinds the signature by computing $s = \tilde{s}^r \bmod n$.

Theorem 2. *Under the hardness of the discrete logarithm problem and the generalized BBS assumption, the scheme above is a (T, t, ϵ) timed RSA signature scheme.*

Proof (sketch). We first prove that the scheme is "binding," i.e., that after successful completion of the commit phase, the receiver is convinced that he will be able to retrieve the correct signature. This will follow from the fact that our time-line commitment is binding (Theorem 1), a proof of uniqueness (the

[7] Using the techniques of [CDS94], some of these proofs can be combined. Here we present them separately for clarity.

same value committed to is used for the signature blinding), and a proof of correct blinding (once retrieved, the value committed to will produce the correct signature).

To see uniqueness, we first establish that the value from the auxiliary commitment, call it r', is the same as the value r hidden by the time-line (step 3 of the commit phase). Note that the proofs that $\log_{g'} b_{\text{aux}} = \log_{b_{\text{aux}}} b$ and INTVL($r_{\text{aux}} \in [0, N-1]$) establish that $r' = {r_{\text{aux}}}^2 \bmod N'$, and hence that $r' \in Q_N$ (the set of quadratic residues modulo N). Then, the proof that $\log_{g'} b = \log_b B$ in turn proves that $R_k = r'^2 \bmod N$, i.e., that r' is the square root of the value $R_k = h^{2^{2^k}} \bmod N$ that the verifier knows. Thus, $r' = r$. Next, proof EQLOG-2(r, n, N') in step 4 of the commit phase in conjunction with INTVL($r \in [0, N-1]$) show that, necessarily, the same value r is used to blind the signature on message m.

For the correct blinding, since it was proven that $\log_X m = \log_{g'} b\,(= r)$ for $X = \tilde{s}^e$, the receiver knows that he will be able to compute s as $s = \tilde{s}^r \bmod n$, once he has computed r. This holds in spite of the fact that r may be larger than the unknown quantity $\varphi(n)$, as this merely results in a "wrap-around."

The receiver will be able to retrieve r, either directly from the signer, or in time T by running the time-line's forced open algorithm. ∎

Third-party release. We see that the committer has to compute $s^{1/r} \bmod n$ in the blinding step (step 1). This corresponds to raising the signature s to the value $1/r \bmod \varphi(n)$. This is not a problem if the committer is also the signer, as he would know $\varphi(n)$. However, the above technique does not allow one party to commit to a signature that was generated by another party. In the full version of the paper, we show how to do this efficiently for RSA signatures. We now show how this can be done for Schnorr and DSA signatures, allowing the timed release of signatures by parties other than the signer. This may be a beneficial property in payment schemes, among other applications.

4.2 Timed Schnorr Signatures

In order not to diverge too much from common notation, but still not to overload parameter names, we will be using an horizontal line above signature-specific names. Let $\overline{g}$ be a generator of a group of size $\overline{q}$, and let all computation, where applicable, be modulo $\overline{p}$, where $\overline{q}$ divides $\overline{p} - 1$. We let $\overline{x} \in \mathbb{Z}_{\overline{q}}$ be the secret key, and $\overline{y} = \overline{g}^{\overline{x}}$ be the public key. Recall that a Schnorr signature is generated as follows. The signer selects $\overline{k} \in_R \mathbb{Z}_{\overline{q}}$, and computes $\overline{\rho} = \overline{g}^{\overline{k}} \bmod \overline{p}$ and $\overline{s} = \overline{k} + \overline{x}h(m, \overline{\rho}) \bmod \overline{q}$, where m is the message to be signed.

Commit phase. We assume for simplicity that the signer and the verifier have already done a time-line commitment TL(r), and that the committer knows a signature $(\overline{\rho}, \overline{s})$ on a message m known by both the committer and the receiver. (Note that the comitter does not need to be the signer in the following.) The committer and receiver perform the following steps:

1. *Application of blinding factor.* Let r be the blinding value time-committed to. Let $\tilde{s} = \overline{s}/r \bmod \overline{q}$ be the blinded signature, and $\tilde{g} = \overline{g}^r \bmod \overline{p}$ the blinded generator. The committer outputs $(\tilde{g}, \overline{p}, \tilde{s})$.
2. *Auxiliary commitments and proof of uniqueness.* This is performed identically to how it was performed for timed RSA release.
3. *Proof of correct blinding.* The verifier checks that $\tilde{g}^{\tilde{s}} = \overline{p}\overline{y}^{h(m,\overline{\rho})} \bmod \overline{p}$. The committer proves that $log_{\tilde{g}}\tilde{g} = log_{g'}b\,(= r)$ using EQLOG-2$(r, \overline{p}, N')$, and that $r = log_{\tilde{g}}\tilde{g}$ is in the right interval using INTVL$(r \in [0, N-1])$.

Open and forced open phases. The verifier obtains r from the signer or from the time-line Then, the verifier unblinds the signature: $\overline{s} = \tilde{s}r \bmod \overline{q}$, and outputs $(m, \overline{\rho}, \overline{s})$.

Theorem 3. *Under the hardness of the discrete logarithm problem and the generalized BBS assumption, the scheme above is a (T, t, ϵ) timed Schnorr signature scheme.*

The proof of this theorem is similar to that for RSA signatures (Theorem 2), and will be given in the full version of the paper.

4.3 Timed DSA Signatures

We use similar notation here as above for Schnorr signatures. Let m be the (hashed) message to be signed. A DSA signature is generated by the signer selecting $\overline{k} \in_R \mathbb{Z}_{\overline{q}}$, computing $\overline{\rho} = \overline{g}^{\overline{k}} \bmod \overline{p}$, $\overline{\lambda} = \overline{\rho} \bmod \overline{q}$, and $\overline{s} = \overline{k}^{-1}(m + \overline{x}\overline{\lambda}) \bmod \overline{q}$. As for Schnorr signatures, the committer may be a different entity than the signer; in that case, however, we assume that he knows the auxiliary information $\overline{\rho}$.

Commit phase. We assume for simplicity that the signer and the verifier have already done a time-line commitment TL(r), and that the committer knows the unreduced signature $(\overline{\rho}, \overline{s})$ on a message m. The committer and receiver perform the following steps:

1. *Application of blinding factor.* Let r be the blinding value committed to. Let $\tilde{\rho} = \overline{\rho}^r \bmod \overline{p}$, and $\tilde{s} = \overline{s}/r \bmod \overline{q}$. The committer outputs $(\tilde{\rho}, \overline{\rho}, \overline{\lambda}, \tilde{s})$.
2. *Auxiliary commitments and proof of uniqueness.* This is performed identically to how it was performed for timed RSA and Schnorr release.
3. *Proof of correct blinding.* The verifier checks that $\tilde{\rho}^{\tilde{s}} \equiv_{\overline{p}} \overline{g}^m \overline{y}^{\overline{\lambda}}$. The committer proves that $log_{\overline{\rho}}\tilde{\rho} = log_{g'}b\,(= r)$ using EQLOG-2$(r, \overline{p}, N')$, and that $r = log_{\overline{\rho}}\tilde{\rho}$ is in the right interval using INTVL$(r \in [0, N-1])$.

Open and forced open phases. The value r is computed as above. Then, the verifier unblinds the signature: $\overline{s} = \tilde{s}r \bmod \overline{q}$. He outputs $(m, \overline{\lambda}, \overline{s})$.

Theorem 4. *Under the hardness of the discrete logarithm problem and the generalized BBS assumption, the scheme above is a* (T, t, ϵ) *timed DSA signature scheme.*

Proof in the full version of the paper.

Acknowledgements. We thank Daniel Bleichenbacher, Ari Juels and Carl Pomerance for many insightful comments and suggestions.

References

[Bao98] F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *Proc. CARDIS'98*, 1998.

[BBS86] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.

[BCDvdG87] E. Brickell, D. Chaum, I. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret (extended abstract). In *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 156–166. Springer-Verlag, 1988, 16–20 August 1987.

[BG96] M. Bellare and S. Goldwasser. Encapsulated key escrow. In MIT/LCS/TR-688, 1996.

[BG97] M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *Proc. ACM CCS*, pages 78–91, 1997.

[Ble00] D. Bleichenbacher. On the distribution of DSA session keys. Manuscript, 2000.

[BN00] D. Boneh and M. Naor. Timed commitments (extended abstract). In *Advances in Cryptology—CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer-Verlag, 2000.

[Bou00] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer-Verlag, 2000.

[BT99] F. Boudot and J. Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *Proc. 2nd International Conference on Information and Communication Security*, volume 1726 of *Lecture Notes in Computer Science*, pages 87–102. Springer-Verlag, 1999.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Plenum Press, New York and London, 1983, 23–25 August 1982.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 21–25 August 1994.

[CEvdG87] D. Chaum, J. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology—EUROCRYPT 87*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. Springer-Verlag, 1988, 13–15 April 1987.

[CFT98] A. Chan, Y. Frankel, and Y. Thiounis. Easy come – easy go divisible cash. In *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer-Verlag, 1998.

[CM99] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes (extended abstract). In *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 414–430. Springer-Verlag, 1999.

[CP92] D. Chaum and T. Pedersen. Wallet databases with observers (extended abstract). In CRYPTO'92 [CRY92], pages 89–105.

[CRY92] *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993, 16–20 August 1992.

[DN92] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In CRYPTO'92 [CRY92], pages 139–147.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987, 11–15 August 1986.

[GMP01] S. Galbraith, W. Mao, and K. Paterson. A cautionary note regarding cryptographic protocols based on composite integers. In HPL-2001-284, 2001.

[Mao98] W. Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Proc. Public Key Cryptography '98*, pages 27–42, 1998.

[May93] T. May. Timed-release crypto. In `http://www.hks.net.cpunks/cpunks-0/1460.html`, 1993.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity. In Jacques Stern, editor, *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[RSW96] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. In MIT/LCS/TR-684, 1996.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985, 19–22 August 1984.

[Sha95] A. Shamir. Partial key escrow: A new approach to software key escrow. In *Key Escrow Conference*, 1995.

Quasi-Efficient Revocation of Group Signatures

Giuseppe Ateniese[1], Dawn Song[2], and Gene Tsudik[3]

[1] Department of Computer Science
The Johns Hopkins University, Baltimore, MD. USA
ateniese@cs.jhu.edu
[2] Computer Science Division, EECS
University of California, Berkeley, CA. USA
dawnsong@cs.berkeley.edu
[3] Department of Information and Computer Science,
University of California, Irvine, CA. USA
gts@ics.uci.edu

Abstract. Several interesting group signature schemes have been proposed to-date. However, in order for the entire group signature concept to become practical and credible, the problem of secure and efficient group member revocation must be addressed. In this paper, we construct a new revocation method for group signatures based on the signature scheme by Ateniese et al. [ACJT]. This new method represents an advance in the state-of-the-art since the only revocation schemes proposed thus far are either: 1) based on implicit revocation and the use of fixed time periods, or 2) require the signature size to be linear in the number of revoked members. Our method, in contrast, does not rely on time periods, offers constant-length signatures and constant work for the signer.

Keywords: Group signatures, revocation of group membership credentials, dynamic groups.

1 Introduction

Group signatures are a relatively new concept introduced by Chaum and van Heijst [CvH91] in 1991. A group signature, akin to its traditional counterpart, allows the signer to demonstrate knowledge of a secret with respect to a specific document. A group signature is publicly verifiable: it can be validated by anyone in possession of a group public key. However, group signatures are anonymous in that no one, with the exception of a designated group manager, can determine the identity of the signer. Furthermore, group signatures are unlinkable which makes it computationally hard to establish whether or not multiple signatures are produced by the same group member. In exceptional cases (such as a legal dispute) any group signature can be "opened" by a group manager to reveal unambiguously the identity of the actual signer. At the same time, no one — including the group manager — can misattribute a valid group signature.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 183–197, 2003.

These features of group signatures make them attractive for many specialized applications, such as voting and bidding. They can, for example, be used in invitations to submit tenders [CP95]. All companies submitting a tender form a group and each company signs its tender anonymously using the group signature. Once the preferred tender is selected, the winner can be traced while the other bidders remain anonymous.

More generally, group signatures can be used to conceal organizational structures, e.g,, when a company or a government agency issues a signed statement. Group signatures can also be integrated with an electronic cash system whereby several banks can securely distribute anonymous and untraceable e-cash. The group property can offer a further advantage of concealing the identity of the cash-issuing bank [LR98].

Several group signature schemes have been developed, e.g., [CP95], [CS97], [AT99a], [CM98a] [Cam97]. Some have been subsequently broken, others are impractical due to long public keys and/or long signatures while most remaining schemes offer uncertain (i.e., unproven) security. One exception is a recent scheme by Ateniese, et al. [ACJT] which is both efficient and provably secure.

Motivation

As observed in [AT99], to be truly useful, a group signature scheme must support dynamic group membership. Current state-of-the-art group signature schemes (such as [CS97], [CM98a], and [ACJT]) support growing membership: new members can join without precipitating changes in the group public key or re-issuing group membership certificates for existing members. However, *shrinking* group membership has not been given the same attention. We believe this is because either it was not deemed important enough, or (more likely) no viable solutions were known.

In many realistic group settings, group members are equally likely to join, leave voluntarily or be excluded, from the group. Therefore, we consider supporting growing and shrinking membership of equal importance. Starting from this premise, we claim that group signature schemes, no matter how elegant or how secure, will remain a neat and curious tool in a theoretical cryptographer's "bag-of-tricks" until a secure and efficient method to support both growing and shrinking membership is found.

Contribution

In this paper, we construct and demonstrate an effective and secure revocation method for group signatures based on the signature scheme by Ateniese et al. [ACJT]. This new method represents an advance in the state-of-the-art since the only revocation schemes proposed thus far are either:[1]

1. Based on implicit revocation, (loosely) synchronized clocks and the use of fixed time periods, or

[1] See Section 2 for details.

2. Require group signature size to be $O(n)$ where n is the number of revoked members and ask the signer to perform $O(n)$ work to compute each signature.

Our method, in contrast, offer explicit (CRL-based) revocation, requires no time periods and offers constant-length signatures and constant work for signers.

Broadly speaking, this paper has but one contribution: it demonstrates the first viable group revocation scheme based on the only provably secure and efficient group signature scheme proposed to-date. At the same time, it should be noted from the outset that the revocation method described in this paper – albeit viable – is not quite practical for reasons to be discussed below. However, we believe that this result moves the field one (perhaps, small) step closer to reality.

Organization. The rest of the paper is organized as follows. We first provide, in the next section, an overview of related work. In Section 3 we discuss some preliminaries. Next, Section 4 discusses revocation issues in the context of group signatures. Section 5, summarizes the Ateniese et al. group signature scheme. Sections 6 overviews a very simple revocation scheme followed by the new quasi-efficient revocation scheme and its informal analysis presented in Sections 7 and 8, respectively. We conclude the paper with the summary and future work.

2 Related Work

Bresson and Stern [BS2000] proposed the first solution for revocation of group signatures. Unfortunately, it requires the signature size to be linear with respect to the number of revoked members. Moreover, it is based on the group signature scheme proposed by Camenisch and Stadler [CS97] which has been found to have certain security problems [AT99] and has not been proven secure even in its modified version.

In a recent paper, Song [Song01] proposed two interesting revocation methods based, like the present work, on the ACJT scheme. (Recall that ACJT is provably secure.) Both methods are notable since – in addition to standard revocation – they also provide retroactive revocation as well as forward security. (In fact, the emphasis is on forward security.) Moreover, they offer constant-length signatures which is an improvement over the Bresson and Stern's result. However, one important feature of Song's methods is the use of fixed (in both length and number) time periods to support revocation. In particular, each member's certificate must evolve in every time period and all verifiers must be aware of this evolution. Also, the maximum number of time periods is fixed and embedded in each member's group certificate. While appropriate for some settings, this solution is not very general since it is hard (in fact, impossible) to revoke a member within a time period. Furthermore, the security of one of the methods is based on a new and uncertain cryptographic assumption which is appreciably stronger than the Decision Diffie-Hellman (DDH) assumption. The second scheme relies on a method (one-way function) of deterministically computing a

fixed-length sequence of prime numbers starting with an initial prime which may be inefficient.

3 Preliminaries

Group-signature schemes are typically defined as follows:[2]

Definition 1. *A* group signature scheme *is a digital signature scheme comprised of the following five procedures:*

`SETUP`: *A probabilistic algorithm which – on input of a security parameter ℓ – outputs the initial group public key $\mathcal{Y}$ (including all system parameters) and the secret key $\mathcal{S}$ for the group manager.*

`JOIN`: *A protocol between the group manager and a user that results in the user becoming a new group member. The user's output is a membership certificate and a membership secret.*

`SIGN`: *A probabilistic algorithm that on input a group public key, a membership certificate, a membership secret, and a message m outputs group signature of m.*

`VERIFY`: *An algorithm for establishing the validity of an alleged group signature of a message with respect to a group public key.*

`OPEN`: *An algorithm that, given a message, a valid group signature on it, a group public key and a group manager's secret key, determines the identity of the signer.*

A secure group signature scheme must satisfy the following properties:

Correctness: Signatures produced by a group member using `SIGN` must be accepted by `VERIFY`.

Unforgeability: Only group members are able to sign messages on behalf of the group.

Anonymity: Given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager.

Unlinkability: Deciding whether two different valid signatures were computed by the same group member is computationally hard.

Exculpability: Neither a group member nor the group manager can sign on behalf of other group members.[3]

Traceability: The group manager is always able to open a valid signature and identify the actual signer. Therefore, any colluding subset of group members cannot generate a valid signature that the group manager cannot link to one of the colluding group members.

In order to provide revocation of membership, an additional property is necessary:

[2] An in-depth discussion on this subject can be found in [Cam98].

[3] However, nothing precludes the group manager from creating phantom signers and then producing group signatures. The same risk occurs with respect to CA-s in traditional (non-group) PKI-s.

Revocability: A signature produced using SIGN by a revoked member must be rejected using a (potentially modified) VERIFY. Equivalently, a signature produced using SIGN by a valid member must be accepted by VERIFY.

The efficiency of of a group signature scheme depends on a number of factors. Usually, the costs of SIGN and VERIFY as well as the sizes of the group signature and the group public key are the most important efficiency measures.

4 Revocation Preliminaries

In general, as soon as a member is revoked, there must be a way to unambiguously determine her revocation status. At the same time, it is sometimes desirable that all signatures generated by a group member before revocation remain valid and secure, i.e., anonymous and unlinkable. This property was first defined in [AT99] and later refined and referred by Song [Song01] as *backward unlinkability*. We observe, however, that a scheme providing backward unlinkability will allow any revoked members to generate signatures in the future and claim that they were generated before revocation occured. This is particularly unpleasant in the context of group signatures where each user signs on behalf of the entire group and signatures are linked to the group's intentions rather than to those of the single signer. Thus, another level of revocation could require to link and identify **all** the signatures generated by a revoked group member. We refer to this revocation flavor as *unconditional linkability*.

One simple way to obtain revocation is to issue a new group public key and new group certificates to all valid members whenever a group member (or a number thereof) leaves or is ejected. However, this would entail a heavy cost and a significant inconvenience. First, all potential verifiers must be notified of the change. This appears to be unavoidable. Second, all remaining members must participate in a JOIN protocol with the group manager. This represents an extra burden for the members since the JOIN protocol is always on-line and usually involves significant effort (as compared to SIGN or VERIFY).

However, it is possible to avoid running interactive JOIN protocols with all members. This can be achieved by generating a new group public key and issuing new group certificates for all members **without** any interaction. As an illustration, we sketch out (in Section 6) a simple method based on the ACJT scheme. (A very similar approach can be constructed with the Camenisch/Stadler group signature scheme [CS97].) However, this approach is not very practical as it involves the issuance of many new membership certificates and requires each group member to fetch its new certificate following every member leave event.

To achieve more effective and efficient revocation, we need to avoid issuing new group certificates to non-revoked members. An ideal revocation method would employ the revocation paradigm commonly used in traditional signature schemes: a verifier simply checks the signer's certificate against the current Certificate Revocation List (CRL). This paradigm is attractive since the signer is unaware of the ever-changing CRL and the revocation checking burden is placed on the verifier. In our setting, however, a group signature always contains in

some form an encrypted version of the signer's group certificate. As pointed out in [ACJT], encryption of the certificate must be semantically secure in order to prevent linking of group signatures.

The very same semantic security makes it impossible for the verifier to link a group signature to a (potentially) revoked group certificate (or some function thereof) that has to appear as part of a CRL. To see why this is the case, consider the opposite: if a verifier is able to link a single group signature to a certain CRL entry, then the same verifier can link multiple group signatures (all by one signer) to the very same CRL entry. This is not a problem if the signer is revoked before all of these group signatures are generated. However, if a verifier can link (based on a current CRL) a revoked signer's signatures computed before revocation, the property of *backward unlinkability* is not preserved. Therefore, we claim that the signer must somehow factor in the current CRL when generating a group signature. In fact, the signer must prove, as part of the signing, that its group certificate (or a function thereof) is not part of the current CRL.

The above is the general approach we take in this paper. The method outlined in detail below (in Section 7) requires a signer to prove non-membership of the current CRL as part of signature generation. The verifier, in turn, checks revocation as part of signature verification. The end-result is that the notion of the group public key is extended to include the latest group CRL.

Revocation Efficiency. We identify the following measures of efficiency or practicality for any revocation method (of course, only in the context of group signatures):

- **Increased Signature Size**: is the most important measure of a revocation method's efficiency. Clearly, signature size should be minimized. More generally, if the underlying group signature scheme has $O(x)$-size signatures (where x is a constant, or some function of group size), revocation checking should ideally not change the signature size in the $O()$ notation.
- **Signer Cost:** is the additional cost of generating a group signature that proves non-revocation of the signer. Ideally, this added cost is constant.
- **Verifier Cost:** is the additional cost of verifying a group signature that proves non-revocation of the signer. As above, this added cost is, at best, constant.
- **CRL Size:** is an essential measure since it effectively determines the overall size of the group public key.
- **CRL Issuance Cost:** is the cost of composing and issuing a new CRL (by the group manager) each time a group member must be revoked. While not completely negligible, this efficiency measure is the least significant of the above.

5 The ACJT Group Signature Scheme

In this section we provide an overview of the ACJT scheme [ACJT]. (Readers familiar with ACJT may skip this section with no loss of continuity.) In

its interactive, identity escrow form, the ACJT scheme is proven secure and coalition-resistant under the Strong RSA and DDH assumptions. The security of the non-interactive group signature scheme relies additionally on the Fiat-Shamir heuristic (also known as the random oracle model).

Let Λ and Γ be two integral ranges as defined in [ACJT] and let $\mathcal{H}$ be a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^k$. The ACJT scheme is defined in the set of quadratic residues QR(n) where $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ (p, q,p',q' are all prime numbers). The group public key is $\mathcal{Y} = (n, a, a_0, y = g^x, g, h)$ where a, a_0, g, h are randomly selected $\in_R$ QR(n). The corresponding secret key (known only to GM) is: $\mathcal{S} = (p', q', x)$.

To join the group, an user engages in a JOIN protocol with the group manager and receives a group certificate $[A_i, e_i]$, where $A_i = (a^{x_i} a_0)^{1/e_i} \bmod n$ with $e_i \in \Gamma$ and $x_i \in \Lambda$ (x_i is known only to the user).

In order to (anonymously) sign a message, the group member has to prove possession of his member certificate without revealing it. In particular, a group member P_i computes:

$$T_1 = A_i\, y^w, T_2 = g^w, T_3 = g^{e_i}\, h^w, SK(m).$$

The value $SK(m)$, computed over a message m, represents a signature of knowledge of (see [ACJT] for details):

1. a value x_i such that $(a^{x_i} a_0)^{1/e_i}$ is the value that is ElGamal-encrypted in (T_1, T_2) under GM's public key y
   and
2. an e_i-th root of that encrypted value, where e_i is the first part of the representation of T_3 w.r.t. g and h, such that e_i lies in Γ.

In case the actual signer has to be subsequently identified (e.g., to settle a dispute) GM opens the ElGamal encryption to reveal the group certificate A_i, unique to each member. In addition, GM provides a proof that:

$$\text{Dlog}_g y = \text{Dlog}_{T_2}(T_1/A_i)$$

which demonstrates that the encryption was opened correctly.

6 Reissuance-Based Revocation in ACJT

We now discuss two simple revocation schemes for ACJT. When a set of members needs to be revoked, both schemes require the group manager to re-issue group certificates to all remaining members. The first scheme concetrates the cost of reissuance in the group manager while a group member does negligible amount of work. The second scheme distributes the work among the group manager and the idividual members.

Scheme I

Recall that the JOIN protocol in the ACJT scheme results in the issuance of a secret membership certificate $[A_i, e_i]$ where $A_i = (a^{x_i} a_0)^{1/e_i} \bmod n$.

Since the group manager is the one choosing each prime e_i at JOIN time, it can easily issue a new certificate to every valid group member without any additional interaction. Specifically, GM can issue a new certificate of the form:

$$A_{k,i} = (a_k^{x_i} a_{0,k})^{1/e_i} \bmod n,\ e_i$$

We use the index k to denote the sequence number of U_i's group certificate; equivalently, k is the number of shrinking membership changes that took place since the U_i joined the group.

The values a_k and $a_{0,k}$ are generated by GM for every update (re-issue) of the group public key. One simple and efficient way of generating these values for GM to select a secret random number $r \in \mathbb{Z}^*_{p'q'}$ and compute $a_k = a_{k-1}^r \bmod n$ and $a_{0,k} = a_{0,k-1}^r \bmod n$. Notice that a revoked user can not obtain a new-issue group certificate since the value r is not known. Furthermore, GM can easily compute the value $a_k^{x_i}$ as $a_k^{x_i} = (a_{k-1}^{x_i})^r$ (the initial value of $a_{k-1}^{x_i}$ can be stored by GM during JOIN).

Next, GM publishes all newly issued certificates in some public forum, e.g., a bulletin board or a web page. Alternatively, it can broadcast the whole batch to the group. Of course, to keep the number of group members secret, GM can (and should) also issue and publish a sufficient number of fake certificates. The new certificates are accompanied by a new group public key:

$$\mathcal{Y}_k = (n, a_k, a_{0,k}, y, g, h)$$

Obviously, group certificates can not be published in cleartext. Instead, each certificate must be individually encrypted and tagged. One possible format is illustrated in Table 6. The purpose of the search tag is to help each member find its new certificate in the table. (Otherwise, a member would have to try decrypting $n/2$ certificates, on the average, before finding its own.) In this example, every new certificate is encrypted (e.g., using Cramer/Shoup [CS98] under a public key provided by each user at JOIN time). Moreover, a pair-wise secret, s_i, is established between GM and each group member during JOIN such that the search tag can be computed as, for example, a MAC (a construct such as HMAC-SHA1 would suffice) of a random value t selected at random by the group manager each time the table is re-issued.

The present method is both simple and secure.[4] Unfortunately, it is inefficient, since – for every leaving or expelled member – GM needs to perform $O(n)$ cryptographic operations to compose the table. Moreover, each member needs to fetch the entire certificate table (containing its new certificate) as well as the new group public key. Note that just fetching one's own certificate is insecure as it would reveal to a potential eavesdropper the ownership of a group certificate.

[4] Of course, its security requires a suitable encryption function; semantic security (e.g., as in El Gamal) is necessary to prevent distinguishability of encrypted certificates.

Table 1. Re-issued Group Certificate Table

Encrypted Certificate	Search Tag (under t)
$E_1(A_{k,i}, e_i)$	$MAC_{s_1}(t)$
...	...
...	...
...	...
$E_n(A_{k,n}, e_n)$	$MAC_{s_n}(t)$

Scheme II

In this section, we propose another simple revocation scheme which off-loads much of the work (required to re-issue a new group certificate) to the individual group members.

Assume the original group public key is (a_0, a), and n members have joined the group with initial group signing certificate (A_i, x_i, e_i) respectively, obtained when joining the group. Let $f := e_1 * \cdots e_n$, and assume all members know the value of f. We show how to delete a member k as follows. First, GM randomly $u \in QR_n$ as a public coin-flipping which ensures that GM cannot determine the value of u as his wish. GM then computes $t := u^{1/(f/e_k)} \mod n$. Then GM publishes a CRL including t, e_k, and the new public key (a'_0, a) where $a'_0 = a_0 * u$. For a member $i \neq k$, when he sees the CRL, he updates his group signing certificate as $A'_i = A_i * t^{s_i}$, where $s_i = f/(e_i * e_k)$. So the new group signing certificate satisfies $A_i'^{e_i} = a'_0 * a^{x_i}$ (note that $a'_0 := a_0 * u$ is the new group public key). For member k, he cannot compute the corresponding A'_k, so he cannot sign messages corresponding to the new group public key.

We need make a few minor changes to the signing and opening procedure when using this revocation scheme. The group public key a'_0 should be included in hashing in step 2.b in signing procedure in [ACJT]. In the open procedure, assume GM needs to open a signature generated using (A', e) with the corresponding group public key (a'_0, a). GM can compute A'_i as in the original open procedure, and then check against the signing certificate (A_i, e_i) for each member authorized for the corresponding group public key to see whether $A' = A_i u^{1/e_i}$. If the formula holds, then the signer is member i.

This revocation scheme has the following advantages. First, signing and verification operations are constant, i.e. independent on the number of members revoked or in the group. Second, no update is necessary when a new member joins the group. If new members join the group between two revoke events, the information about the newly joined members, namely the $e'_j s$ of the newly joined members can be accumulated and included in the CRL for the later revoke events. GM only needs to do one exponentiation for each update. When several members are revoked at the same time, it is easy to see that the revocation can be consolidated as one update event. The downside is that when deleting a member, each remaining member needs to do an update that is linear to the number of remaining members.

7 CRL-Based Revocation

We begin by assuming, as usual, that a CRL is a structure available at all times from a number of well-known public repositories or servers. A CRL is also assumed to be signed and timestamped by its issuer which can be a universally trusted CA, a group manager or some other trusted party.

In addition to the usual components of a group signature scheme (`SETUP`, `JOIN`, etc.) we introduce an additional algorithm called `REVOKE`. Also, as can be expected, revocation influences `SIGN` and `VERIFY` algorithms. The `JOIN` and `OPEN` components remain unchanged. The only (addition) change in `SETUP` is as follows:

`SETUP (new step):`

- Select $\bar{G} =< \bar{g} >$ of order n in which computing discrete logarithms is hard. For example, $\bar{G}$ can be a subgroup of $Z^*_{\bar{p}}$ for a prime $\bar{p}$ such that n divides $(\bar{p}-1)$.

The new `REVOKE` algorithm shown below is executed by the group manager whenever a member (or a collection of members) leaves or is expelled. (Note that `REVOKE` may also be executed as a "decoy", i.e., without any new membership revocation activity.) The cost to the GM is linear in the number of revoked group members.

`REVOKE:` (We use s to denote the index of the current CRL issue.)

1. First, choose a random element $b_s \in_R \mathrm{QR}(n)$ (of order $p'q'$). b_s becomes the current revocation base.
2. WLOG, assume that m users: $U_1, ..., U_m$ are to be revoked.
3. For each revoked U_j, $1 \leq j \leq m$ compute:

$$V_{s,j} = b_s^{e_j}$$

4. The actual revocation list is then published:

$$CRL_s = b_s, \{V_{s,j} \mid 0 < j < m+1\}$$

In the amended `SIGN` algorithm, as part of step 1, member U_i generates two additional values:

$$T_4 = f = \bar{g}^r \text{ where } r \in_R Z_n$$

$$T_5 = f^{b_s^{e_i}} \bmod n$$

U_i then proves, in zero knowledge, that the double discrete logarithm of T_5 with bases f and b_s, respectively is the same as the discrete logarithm of T_3's representation base g. Since T_3 is computed as $g^{e_i} h^w \bmod n$, the resulting proof of knowledge (SKLOGEQLOGLOG) is verifiable if and only if the same e_i is

used the construction of both T_5 and T_3. The details of this proof are presented below.

Remark: the current CRL value b_s must be signed as part of the message "m" which serves as input to the hash function in the actual signature-of-knowledge. This commits the signer to a specific CRL epoch.

In the amended **VERIFY** algorithm we introduce a new steps 3 and 4:

3. For each $V_{s,j} \in CRL$, check if:

$$T_5 == T_4^{V_{s,j}} \bmod n$$

4. Check SKLOGEQLOGLOG, the proof of equality of double discrete logarithm of T_5 and discrete logarithm of T_3's representation base g.

The intuition behind this scheme is straight-forward: if a member U_i is revoked, $V_{s,i}$ is published as part of the current group CRL. Thereafter, in order to produce a group signature, U_i needs to prove that $(b_s)^{e_i}$ does not appear on the CRL which is impossible since $(b_s)^{e_i} = V_{s,j}$ for some j if U_i is revoked.

We claim that the new scheme provides *backward unlinkability* because signatures produced by a revoked user prior to revocation in earlier CRL *epochs* can not be linked to those produced after revocation. Suppose that an adversary is able to link a pre-revocation signature to a post-revocation signature. Then, she can only do so with the help of the *new* values: T_4 and T_5. (Otherwise the ACJT scheme is insecure). Since $T_4 = f$ is chosen at random for each signature, the only way the adversary can link two signatures is using $T_5 = f^{b_s^{e_i}}$. However, this is impossible since the respective b_s values are different and unrelated for any pair of signatures computed in different CRL epochs.

To be more specific, we need to consider two cases: linking two signatures from different CRL epochs and linking two signatures from the same CRL epoch. It is easy to see that both are computationally infeasible assuming DDH is hard. In more detail, it is easy to see that the former is infeasible for some $T_5^1 = f'^{b_s'^{e_i}}$ and $T_5^2 = f''^{b_s''^{e_i}}$ where $f' \neq f$" and $b_s' \neq b_s$". The latter is also infeasible for some $T_5^1 = f'^{b_s^{e_i}}$ and $T_5^2 = f''^{b_s^{e_i}}$ where $f' \neq f$", based on a well-known variant of the DDH problem.

Achieving Unconditional Linkability. The values T_6 and T_7 defined above also give the group manager the flexibility of selecting the desired revocation flavor. More specifically, the group manager can decide to revoke all the signatures of a particular member, thus achieving unconditional linkability as defined in Section 4. To do so, it is sufficient to release as part of the CRL the value e_i, rather than $b_s^{e_i}$, thus allowing a verifier to check (for any signature) whether $T_6^{e_i}$ is equal to T_7.

Obscuring CRL Size. Over time, the size of the CRL may leak some information about the population of the group: by observing long-term changes and fluctuations in the size of the CRL, an adversary can guess the number of group members. For this reason, it may be necessary to obscure the true CRL size. This can be done by introducing a number of fake (but well-formed) CRL entries.

Proofs Involving Double Discrete Logs. Proofs of knowledge of double discrete logarithms have been used in the past. Examples include Stadler's technique for publicly verifiable secret sharing [Stad96] and Camenisch/Stadler group signature scheme and its derivatives [CS97,LR98,KP98]. All of the these involve only proofs of knowledge; whereas, in the above scheme, we need a new proof (SKLOGEQLOGLOG) of equality of a double discrete log and a (single) discrete log of a representation. The technique we use is a minor variation of the SKLOGLOG (proof of knowledge of double discrete logarithm) proposed by Camenisch [Cam98], Stadler [Stad96] and Camenisch/Stadler [CS97]. The proof is constructed as follows:

Given $y_1 = g^{a^x}$ and $y_2 = g_1^x g_2^w$, we want to prove that:

$$\mathrm{Dlog}_a(\mathrm{Dlog}_g y_1) = \mathrm{Dlog}_{g_1}(y_2/g_2^w) \ \ (= x)$$

Let $\ell \leq k$ be two security parameters and $H : \{0,1\}^* \longrightarrow \{0,1,\}^k$ be a cryptographic hash function. Generate 2ℓ random numbers $r_1, \ldots, r_\ell$ and $v_1, \ldots, v_\ell$. Compute, for $1 \leq i \leq \ell$, $t_i = g^{a^{r_i}}$ and $t'_i = g_1^{r_i} g_2^{v_i}$. The signature of knowledge on the message m is $(c, s_1, s_2, \ldots, s_\ell, s'_1, s'_2, \ldots, s'_\ell)$, where:

$$c = H(m||y_1||y_2||g||a||||g_1||g_2||t_1|| \cdots ||t_\ell||t'_1|| \cdots ||t'_\ell)$$

and

$$\textbf{if } c[i] = 0 \textbf{ then } s_i = r_i, s'_i = v_i;$$
$$\textbf{else } s_i = r_i - x, s'_i = v_i - w;$$

To verify the signature it is sufficient to compute:

$$c' = H(m||y_1||y_2||g||a||g_1||g_2||\bar{t}_1|| \cdots ||\bar{t}_\ell||\bar{t}'_1|| \cdots ||\bar{t}'_\ell)$$

with

$$\textbf{if } c[i] = 0 \textbf{ then } \bar{t}_i = g^{a^{s_i}}, \bar{t}'_i = g_1^{s_i} g_2^{s'_i};$$
$$\textbf{else } \bar{t}_i = y_1^{a^{s_i}}, \bar{t}'_i = y_2 g_1^{s_i} g_2^{s'_i};$$

and check whether $c = c'$.

8 Efficiency Considerations

The new revocation scheme presented in Section 7 is quasi-efficient in that a group signature is of **fixed size** and a signer performs a **constant** amount of work in generating a signature. This is, as claimed earlier, an improvement on prior results. However, proofs involving double discrete logs are notoriously expensive. For example, if we assume a hash function $H : \{0,1\}^* \longrightarrow \{0,1,\}^k$ where $k = 160$ bits (as in SHA-1), and we assume that the security parameter

$\ell = k$, then each SIGN operation will take approximately 500 exponentiations. The cost of VERIFY is roughly the same. Moreover, with a 1024-bit modulus, a signature can range into hundreds of Kbits. This is clearly not efficient.

Remark: one way to reduce the costs of SIGN and VERIFY and (roughly) halve the number of exponentiations is to ask the signer to release as part of SIGN two additional values: $T_6 = \hat{g}$ and $T_7 = \hat{g}^{e_i}$ for a randomly chosen $\hat{g} \in QR_n$. The signer would show that T_7 is correctly formed and then prove the equality of the discrete log base $\hat{g}$ of T_7 and the double discrete log of T_5 (base f and b_s, respectively). This proof would be both shorter and less costly then the proof of equality of double discrete log (of T_5) and discrete log of representation of T_3.

Despite the usage of double discrete logarithm proofs and in contrast with Bresson and Stern's scheme [BS2000], the cost of SIGN in our scheme is constant (independent of group size of number of revoked members) and signatures are of a fixed size. Comparing with Song's schemes [Song01], our scheme is more expensive for both SIGN and VERIFY due to the double discrete log proof. One advantage of our scheme is in not using fixed (in length and number) time periods. Consequently, a new revocation list can be issued at any time. Also, we introduce no new cryptographic assumptions. Song's two schemes, however, have the benefit of *retroactive public revocability* meaning that a member's signatures can be revoked for one or more **past** time periods. This is a feature that our method does not offer.

The cost of REVOKE in our scheme is linear in the number of revoked members: GM performs one exponentiation for each CRL entry $V_{s,j}$. This is comparable with prior results in both [BS2000] and [Song01] schemes.

9 Summary and Future Work

We presented a new revocation method for group signatures based on the ACJT signature scheme [ACJT]. The new method is more practical than prior art due to fixed-size signatures and constant work by signers. On the other hand, it requires the use of proofs-of-knowledge involving double discrete logs which results in hundreds of exponentiations per signature. Consequently, revocation in group signatures remains inefficient while the following issues remain open:

- Shorter CRL: in our method a CRL is proportional to the number of revoked members. An ideal scheme would have a fixed-size or, at least, a shorter CRL (e.g., logarithmic in the number of revoked members).
- More efficient VERIFY: the cost of VERIFY is linear in the number of revoked members. It remains to be seen whether a constant- or sublinear-cost VERIFY can be devised.
- Double discrete log: proofs using double discrete logarithms are inefficient, requiring many exponentiations. For revocation to become truly practical, we need to devise either more efficient double discrete log proofs or different revocation structures that avoid double discrete log proofs altogether.

Very recently, Camenisch and Lysyanskaya [CL02] developed a revocation technique which improves upon previous work since the verification phase requires constant work. Their technique – based on dynamic accumulators – is quite general and applicable in settings other than just group signatures. However, it is not a fully satisfactory solution for the following reasons:

- The group public key must change even when users are added, whereas, the original group signature scheme avoids this burden.
- The group manager has to maintain two public (and possibly long) lists of values. The first (add-list) is linear in the number of all members added during the lifetime of the group (current as well as revoked) and the second (delete-list) is linear in the number of all revoked users. These lists are not technically part of the public key since the verifier does not need them in order to verify the correctness of a signature.
- Each current group member must have up-to-date add- and delete-lists in order to sign. Also, as part of signing, a member has to perform computation linear in the number of membership changes since the last time he signed. In the worst case, this corresponds to the number of all current as well as all revoked members, which can make signing quite inefficient.

Acknowledgements. We thank the anonymous referees for their comments on the early draft of this paper.

References

[ACJT] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical & Provably Secure Coalition-Resistant Group Signature Scheme In *Advances in Cryptology — CRYPTO '00*, Springer-Verlag, 2000.

[CP95] L. Chen and T. P. Pedersen. New group signature schemes. In *Advances in Cryptology — EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 171–181, 1995.

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424, Springer-Verlag, 1997.

[Stad96] M. Stadler. Publicly Verifiable Secret Sharing, In *Advances in Cryptology — EUROCRYPT '96*, Springer-Verlag, 1996.

[Cam98] J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. PhD thesis, vol. 2 of *ETH Series in Information Security an Cryptography*, Hartung-Gorre Verlag, Konstanz, 1998. ISBN 3-89649-286-1.

[CvH91] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265, Springer-Verlag, 1991.

[KP98] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology — CRYPTO '98*, vol. 1642 of *LNCS*, pp. 169–185, Springer-Verlag, 1998.

[LR98] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography (FC '98)*, vol. 1465 of *LNCS*, pp. 184–197, Springer-Verlag, 1998.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '98*, vol. 1642 of *LNCS*, pp. 13—25.

[AT99a] G. Ateniese and G. Tsudik. Group Signatures a' la Carte. In Proceedings of 1999 ACM Symposium on Discrete Algorithms, ACM Press, 1999.

[AT99] G. Ateniese and G. Tsudik. Some open issues and new direction in group signatures. In Proceedings of 1999 Financial Crypto. Springer-Verlag, 1999.

[BS2000] E. Bresson and J. Stern. Efficient Revocation in Group Signatures, In Proceedings of Public Key Cryptography (PKC'2001), Springer-Verlag, 2001.

[Song01] D. Song. Practical Forward-Secure Group Signature Schemes. In Proceedings of 2001 ACM Symposium on Computer and Communication Security. November 2001.

[Cam97] J. Camenisch. Efficient and Generalized Group Signatures. In Advances in Cryptology - EUROCRYPT '97, vol. 1233 of *LNCS*, pp. 465-479, Springer Verlag, 1997

[CM98a] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology — ASIACRYPT '98*, vol. 1514 of *LNCS*, pp. 160–174, Springer-Verlag, 1998.

[CL02] J. Camenisch and A. Lysyanskaya. Efficient revocation of anonymous group membership certificates and anonymous credentials. IACR eprint archive n. 2001/113.

The Dark Side of Threshold Cryptography

Shouhuai Xu[1] and Moti Yung[2]

[1] Laboratory for Information Security Technology
Department of Information and Software Engineering
George Mason University, Fairfax, VA 22030, USA
sxu1@gmu.edu
[2] Certco, New York, NY, USA
moti@cs.columbia.edu

"The whole is more than the sum of its parts"

Abstract. It is typical for a cryptographic technology to be useful in its primary goal and applications, yet to exhibit also a dark side, namely to allow abuses in some other situations. Examples are subliminal channels in strong (randomized) signature schemes, employing authentication for encryption, kleptography exploiting strong randomness, etc. *Threshold cryptography* was introduced to realize better security and availability. However, its "dark side" has never been addressed seriously. We investigate some possible abuses of threshold cryptography which result from users not possessing the entire private key due to threshold splitting. This is a deficiency which can hurt de-commitment in procedures like "contract signing" and nullify non-transferability properties. To attempt solving the problem, one may suggest to assure that the user has full control of his private key via a zero-knowledge confirmation. However, advances in cryptography itself defeat this, since the Completeness Theorem for secure computations implies that servers in possession of shares of a key can answer on behalf of the "virtual holder" of the entire private key, without compromising the secrecy of their shares. We are then forced to look at more physical limitations of the setting. We propose a notion we call Verifiable Secret Non-Sharing (VSNS) where we can replace the strong (i.e., less realistic) physical isolation assumption (namely, a Faraday cage) with a more realistic *timing* assumption. We then introduce a new class of "combined software engineering and cryptography" adversarial capability, which employs software preprocessing and cryptography in breaking all previous suggestions to our problem. It seems that the adversary is so powerful that we have to rely on certain tamper-resistant device to block it. We show how to prevent a malicious participant from compromising the secrecy of the provers' secret keys in this case. Our treatment is a step towards a model of computing with trusted and non-trusted tamper-proof (black box) elements.

Keywords: contract signing, non-transferable proofs, undeniable signature, abuse of protocols, threshold cryptography, non-sharing, CA, key certification

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 198–219, 2003.

1 Introduction

Threshold cryptography (see [DF89,DF91]) distributes a key among a plurality of servers and requires a quorum of servers to sign/decrypt. It has become a major tool in realizing better security (i.e., more than a single server need to be compromised) and availability (i.e., "single point of failure" devoid). The potential "dark side" of threshold cryptography (e.g., how and when one can abuse it) has not been addressed thoroughly. A source of potential abuses is the fact that third parties do not know whether and how a key is shared due to the transparent nature of the sharing.

The problem is interesting because it is absolutely (i.e., provably) impossible to resolve the sharing issue by a user proving (via traditional protocols and message exchanges) that he knows the key in its entirety. This general obstacle is a result of the Completeness Theorem of secure distributed computation [Y82, GMW87] where users sharing an input can always produce a joint output without revealing the private inputs to each other. In spite of this, we are able to deal with it in the context of designated verifier proofs [JSI96], contract signing [GJM99], and chameleon signatures [KR97] by imposing certain operational restrictions on the model.

1.1 The Problem

The notion of *designated verifier proofs*, due to Jakobsson et al. [JSI96], was designed to cope with "transferability abuse" [DY91, J94] of the verification process in the undeniable signature schemes of Chaum et al. [CvA89]. Specifically, Alice (the prover) proves the statement "either Θ (technically, the corresponding proof of the predicate Θ) is true, or I am Bob" instead of proving Θ directly. Upon seeing such a proof, Bob (the verifier) certainly trusts that Θ is true yet Cindy has no reason at all to believe it. (Retrospectively, the non-transferability of such a proof originates from the *chameleon commitment* schemes introduced in Brassard et al. [BCC88].) Along this way, Garay et al. [GJM99] introduced a new type of digital signature (i.e., *private contract signature*) based on which they proposed an optimistic contract signing protocol with the so-called abuse-freeness property. Abuse-freeness is critical to the fairness of contract signing protocols, yet all of the previously proposed practical solutions are not abuse-free [GJM99]. Also, Krawczyk et al. [KR97] proposed the notion of *chameleon signature* schemes that provides an undeniable commitment of the signers to the contents of the signed documents (as regular digital signatures do) but, at the same time, does not allow the receiver of the signature to disclose the signed contents to any other party without the signer's consent (i.e., non-transferability). Technically, this property comes from the underlying *chameleon hash* [KR97] functions which can be implemented using either claw-free pairs of trapdoor permutations [GMR88] or chameleon commitment schemes [BCC88].

The problem we deal with has not escaped the designers. In fact, it has been explicitly [JSI96, GJM99] and implicitly [KR97] mentioned that the assumption "the secret key of the signature receiver is known to himself" is vital to the usefulness of those proposals. For example, if the receiver Bob holds a pair of private

and public keys (x, g^x) in the settings of [JSI96, KR97, GJM99] such that x is cooperatively generated by Cindy and himself, the intended non-transferability (e.g., of the chameleon hash) or abuse-freeness is completely broken. This is so since seeing the proof "either Θ is true, or I am Bob" is enough to convince Cindy "Θ is true" since Bob himself (without her help) is unable to present the proof "I am Bob". Independently of the importance of the applications, it is really hard to make sure one did not share his private key with any other party in a general sense (ironically, due to the advances of distributed cryptography itself). Consequently, the colluding provers are able to succeed in convincing the verifier in *any* cryptographic protocol (e.g., zero knowledge proof, digital signature, decryption, commitment) that the intended secret key is not shared.

So far, the only two potential approaches to dealing with this problem, as far as we know, are due to [JSI96]:

1. When Bob is registering his public key to have it certified, he has to prove knowledge of his secret key to the Certificate Authority, in a setting where he can only communicate with the CA (e.g., a smart-card setting).
2. When registering his public key, Bob presents his secret key to the CA, who then has to be trusted to neither divulge nor prove knowledge of it to anyone else.

It is even reiterated in [GJM99]: The only defense against such an "ultimate" attack, apart from mere social constraints that make very careful planning and administering of attacks "from the beginning of time" less likely to take place, is to make the certification process including a proof of the knowledge for the to-be-certified party's secret key in an "isolated" manner. This naturally invokes the problem of the existence of such an isolated environment (e.g., what is known in the literature as the Faraday cage [BBD+91]). Unfortunately, the existence of Faraday cage may be problematic because it is based on a controversial and hard to achieve *physical* (instead of *cryptographic*) assumption. Therefore, a lot of open questions are left to be addressed: (1) Can we replace the physical assumption of the existence of Faraday cage with a cryptographic counterpart? (2) Are the two potential solutions proposed in [JSI96] really secure? (3) Can we technically prevent such an ultimate attack? As we shall see, we focus on addressing these questions.

1.2 Our Contributions

We present a cryptographic construction to replace the physical assumption of the existence of Faraday cage [BBD+91]. The function of the newly introduced notion, namely Verifiable Secret Non-Sharing or VSNS for short, is to ensure that the secret key of the certificate holder is not shared with any other party (say, Cindy). The primitive VSNS is based on a new (i.e., the positive use of) *timing* assumption for "secure evaluation of circuits" [Y82, GMW87], which may trigger the research in pursuing their theoretic lower bounds. The timing assumptions combined with the zero-knowledge proof procedure are the crux of this primitive.

We then investigate a harder case and introduce a new class of "combined software engineering and cryptography" attack in which the cheating provers

(say, Bob and Cindy) make use of their knowledge to cooperatively *code and debug* a software (or hardware) module which is thus trusted by them (e.g., to accept their secret inputs and random inputs). The module may further be embedded into certain hardware system (e.g., a smart card). This (abuse employing the existence of a trusted third party) can be done before the start of any protocol. Such a powerful adversary will defeat the two potential approaches proposed in [JSI96] and even the solution in [BC93] for disrupting the man-in-the-middle attack. In other words, even if we assume the existence of Faraday cage and that the CA is trusted not to leak any information about the secret keys of the users, the dishonest provers can still succeed in cheating any verifier. In spite of this, we propose a practical solution based on the existence of tamper-resistant hardware that is secure even in the newly introduced adversarial model. The crux of this solution is isolation of trusted components and careful division of trust between parties, based on which we can technically prevent certain participants from compromising the secrecy of the provers' secret keys.

Remark 1. The main motivation behind this paper is to block the transferability resulted from sharing of certain secrets at the very beginning of system initialization. One may think that transferability is always possible (in small groups) if Bob and Cindy sit together while, for example, Bob receives a proof that "either I am Alice or I can sign m as Bob". However, this is not true. Bob can easily cheat Cindy into accepting a faked proof as follows. Suppose Bob has a brother Bob* whom Bob completely trust, and to whom he gives his private key. Then, Bob asks Cindy to sit near him and see the interaction with the claimed Alice who is in fact impersonated by Bob*. Bob* can always present the proof "either I am Alice or I can sign m as Bob" since he knows Bob's private key. Since Cindy is also smart, she has no reason to be convinced that Alice is at the other end of the communication line. Note that in the application settings we consider here (i.e., designated verifier proof, contract signing, chameleon signatures), Alice does not provide any signature which is non-repudiated in the traditional sense. (In the context of contract signing, Alice only sends her signature after she has been assured that she will also obtain the corresponding signature of Bob.) Moreover, even if Bob's private key is stored in his smart card (i.e., unknown to him), he can ask the application program in the smart card to generate a set of proofs like "either I am Alice or I can sign m as Bob" and can then give them (instead of his private key) to Bob*. Therefore, transferability by having Bob and Cindy sit together is not at all sound in all situations.

1.3 Related Work

Though our usage of *timing* considerations is different from those in the literature [BD90, BC93, DNS98, K96], there do exist certain correlations. In both [BD90] and [BC93], the authors proposed some solutions (based on *accurate* timing) to blocking man-in-the-middle attacks. For example, the basic idea underlying the "distance bounding" protocol of [BC93] is to let the receiver "rapidly" respond to a challenge. As long as each bit of the prover is sent out "immediately" after seeing the challenge bit from the verifier, the delay of the response enables

the verifier to compute an upper-bound on the distance. Another use of *timing* assumption (for certain level of synchronization) is to ensure zero-knowledge under concurrent executions [DNS98]. However, they have no consideration of isolation of provers as in our setting. It should also be noted that the *timing* attack introduced in [K96] is a negative use of timing information.

In timing protocols where users may collaborate in the background (e.g., via preprocessing), other issues must be considered. For example, in the Publicly Verifiable Secret Sharing (PVSS) [S96] replies based on the homomorphism property of the function are used. However, such properties suggest shortcuts for distributed computing. Specifically, the cheating Bob and Cindy can share the random numbers as $k = k_B + k_C$, $w = w_B + w_C$ (refer to [S96] for details). The ElGamal encryption of the secret key $x = x_B + x_C$ can be done in the following way:

$$\begin{aligned} x \cdot y^k &= (x_B + x_C) \cdot y^{k_B + k_C} \\ &= x_B \cdot y^{k_B} + x_C \cdot y^{k_C} + x_B \cdot y^{k_C} + x_C \cdot y^{k_B}. \end{aligned}$$

Thus, we need to avoid such shortcuts which enable "fast collaboration".

The work in [BN00] implemented abuse-freeness adopting an approach (different from [GJM99]) they call "timed commitments". Technically, they allow two parties to fairly exchange RSA or Rabin signatures. Note that our solutions will directly enhance the security of the proposals in [JSI96, KR97, GJM99, GM99].

1.4 Outline

Section 2 is the description of the tools we use. The new primitive Verifiable Secret Non-Sharing (VSNS) is defined and constructed in section 3. Section 4 introduces a class of "combined software engineering and cryptography" attacks, whereas our tamper-resistant hardware-based solution is presented in section 5. We conclude in section 6.

2 Preliminaries

In this paper we work in the standard setting of discrete log-based cryptosystems. Let q be a large prime, $p = lq + 1$ also a prime where $(l, q) = 1$ (e.g., $l = 2$), $g \in_R Z_p$ an element of order q, and G_q the group generated by g. We omit the moduli if they are clear from the context. Let $x \in_R Z_q^*$ be Bob's secret key corresponding to his public key $y = g^x \ mod \ p$.

We work in the Random Oracle model [BR93]. For convenience, a random oracle f is a map that maps any string to an infinite string chosen by selecting each bit of $f(x)$ uniformly and independently, for every x. Since we never use infinitely long output, this is just for the sake of convenience not to specify the concrete output length. Therefore, we have

$$\begin{aligned} &\Pr[f(a+b) = C] \\ &= \Pr[f(a+b) = C | f(a) = A] \\ &= \Pr[f(a+b) = C | f(b) = B], \end{aligned}$$

where + could be addition over the natural numbers N rather then in the group (for example) Z_q from which a and b are chosen. When the random oracle is instantiated from cryptographic hash functions as in [BR93], $f(\cdot)$ can be practically computed (i.e., we explicitly distinguish "practical computations" from "efficient computations") provided that the input is known. However, if Bob and Cindy want to compute securely (in the sense of [Y82, GMW87, G98]) $f(a+b)$ where $a+b$ is unknown to anyone of them, they have to perform a impractical computations, namely they have to compute distributedly while communicating a traversal of the corresponding circuits of f, gate-by-gate and there are no algebraic shortcuts in the computation (i.e., this is polynomially efficient, yet impractical computation in our context).

3 Verifiable Secret Non-sharing (VSNS)

Next we develop a primitive combining zero-knowledge proof of knowledge and timing constraints. It is intended to replace the strong physical assumption of the existence of a Faraday cage.

3.1 The Model

The Participants. Basically and ideally, there are two participants, where the prover Bob intends to apply a "not shared" certificate from the verifier CA. A "not shared" certificate for one's public key g^x is the same as a normal certificate except that it additionally indicates "the private key x of the certificate holder is not shared by the holder with anyone else" (i.e., it is known to the holder). Such a certificate will indicate to Bob's business partners about his suitability while doing business (like contract signing [GJM99, GM99]) with him. In order to do this, Bob needs to convince the CA that his secret key is known to himself. All participants are supposed to be of probabilistic polynomial time capability.

The Communication. All communication channels are public and thus subject to eavesdropping by any party. This is the same as in an interactive zero-knowledge system.

The Adversary. The dishonest Bob's goal is to obtain a "not shared" certificate for his public key g^x corresponding to his secret key x which is cooperatively generated and thus shared by Cindy and himself. Cindy and Bob cooperate in the proof process with the intention to convince (i.e., succeed in cheating) the verifier CA that Bob's secret key is known to him only. However, we do assume that Bob will not let Cindy have complete control over his private key x (i.e., self-enforcement).

The Timing Assumption. Let δ and τ be the upper-bound and the lower-bound for the *one-way* communication delay between the prover and the verifier,

respectively, γ be the upper-bound for the time complexity of the typical computation of function $f(\cdot)$ when the input is not shared by participants, α be the lower-bound for the time complexity of the secure gate-by-gate evaluation of the circuit(s) when the input is shared by some participants. The timing assumption means that $\alpha > \gamma + 2(\delta - \tau)$.

Remark 2. Whether to use the CA off-line (i.e., we trust certificates indicating not-shared private keys corresponding to the certified public keys) or on-line (i.e., we trust third party verification) is a matter of the setting. Since not everyone uses his/her certificate to do business as in [JSI96, KR97, GJM99, GM99], it is completely up to the customer whether or not to apply a "not shared" certificate. Indeed, some users may prefer to share their secret keys with their friends for the sake of better security and availability (e.g., using threshold cryptography). Therefore, this can be viewed as a value-added service of the CA. Technically, either an optional field in the certificate itself (i.e., the syntax approach) or a different certificate signing/verification key (i.e., the semantics approach) can be used to discern whether or not a certificate is a "not shared" one.

Remark 3. The reason we need the timing assumption is that Bob has to cooperate with Cindy in computing the answers to the verifier CA's challenges. The intuitive idea is to make positive use of the time complexity in secure multi-party computation [G98]. For example, let $f(\cdot)$ be a publicly specified function instantiation of the random oracle [BR93], $f(x)$ can be done (say) within 1 second for any input x in the domain. If the input x is shared between Bob and Cindy (e.g., $x = x_B + x_C$), in order to securely (in the sense of [G98]) calculate $f(x)$, they have to traverse the corresponding circuit(s) gate-by-gate with an assumed time complexity (say 30 minutes). Therefore, Bob and Cindy cannot answer the challenges from the verifier rapidly enough. Moreover, the existence of publicly known time bounds for communication is used to ensure that the time difference in the two computations will not be covered by the time difference between the two communication delays. Consequently, if $\alpha - \gamma$ is large enough, the communication upper and lower bounds need not to be very *accurate* (in contrast with [BD90,BC93] which require more refined time management).

3.2 The Objective

For the sake of simplicity, we consider the two-party case (i.e., the cheating provers Bob and Cindy are trying to convince the honest verifier CA) whereas the extension to multi-party is trivial.

Definition 1. *A VSNS is a protocol for the prover Bob to convince the verifier that he knows the secret key corresponding to his public key (alternatively, he does not share his secret key with any third party Cindy). If Bob is cheating, then Cindy has to be involved in the verification process. A VSNS protocol is secure if it is:*

- **complete**: *Any honest prover can always succeed in convincing the verifier by himself without violating the timing constraints.*

- **sound**: *No matter what strategy the cheating provers use, if it acts on its own and the verification is intended not to leak the shared secret key, the probability for the honest verifier to accept within the timing constraints is negligible with respect to the security parameter k. In other words, if the verifier CA accepts, then the secret key is known to one of the cheating provers (either before or after the verification process).*
- **proof-of-knowledge**: *If the probability of accept is large enough, then with almost the same probability there is an extractor which can get the private key by making use of the prover as a black box.*
- **zero-knowledge**: *The protocol is simulatable without access to the secret.*

The above concisely presented definition extends regular zero-knowledge proof of knowledge (taking into account timing constraints) by capturing the idea that if the verifier is convinced then at least one (of Bob and Cindy) knows the secret key in its entirety. Combining with the self-enforcement assumption, we conclude that a successful execution of the protocol means that Bob knows the secret key.

3.3 The Timing Assumption Based VSNS

The protocol is depicted in Figure 1. The intuitive idea is to force the provers to online compute $f(u + v mod\ q)$ and $f(x + u + v mod\ q)$. (i.e., We assume that computing the function twice is done in the protocol). If the prover is honest, the verifier will not quit the protocol due to time-out.

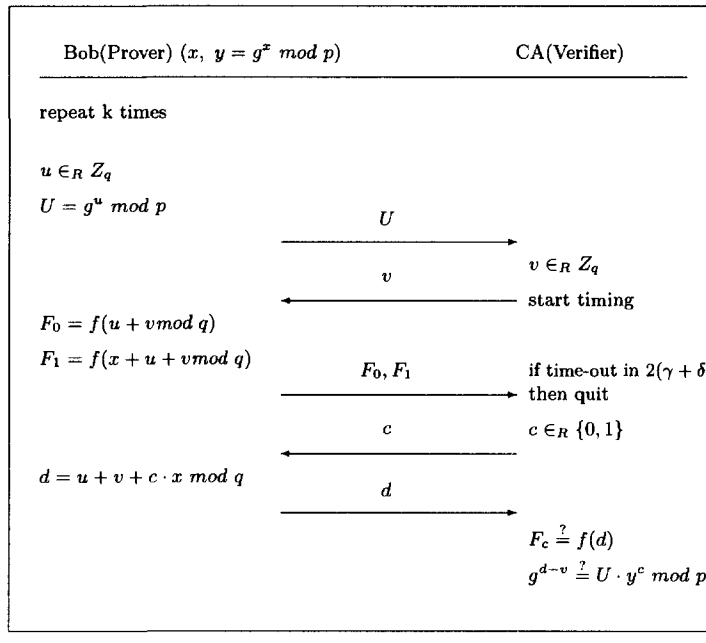


Fig. 1. The VSNS protocol for the proof of the "not shared" knowledge x

3.4 Security Analysis

Theorem 1. *The VSNS protocol is secure.*

Proof (sketch)
Complete. This is trivial from the protocol. Remember δ is the upper-bound for one-way communication delay, γ the upper-bound for the time complexity of typical computation of function $f(\cdot)$ when the input is not shared. Therefore,due to the *timing* assumption we will not timeout since the prover is honest and thus able to present the answer within time $2(\gamma + \delta)$ starting from the point that the verifier sends its message v.

Sound. In order to prove "if the verifier CA accepts, then the secret key is known (either before or after the verification process) to at least one of Bob and Cindy" (combining with the self-enforcement assumption, which also means that Bob knows his secret key), we prove that if Bob and Cindy share his secret key (which means that the verification process preserves this secrecy property), then the verifier CA will reject except with negligible probability.

First, the probability for anyone not knowing the secret x to convince the verifier is at most 2^{-k}. (The standard two-challenge zk-proofs where without knowing the secret, the cheater can prepare to answer only one challenge.)

Second, consider sharing: suppose $x = x_B + x_C$ and $u = u_B + u_C$, where (x_B, u_B) are the shares held by Bob, and (x_C, u_C) are the shares held by Cindy, respectively. There are two cases.

- They only compute and present one of F_0 and F_1 (e.g., by computing $u+v = u_B+u_C+v$ and then $f(u+v)$), and choose a random garbage for the other. In this case, they can present F_0 and F_1 before the protocol time-out. However, the probability for them to succeed in cheating the verifier is at most 2^{-k} ($1/2$ per iteration, as mentioned above).
- They compute and present correct F_0 and F_1. There are only three strategies in realizing this. One of these three is the majority strategy (used at least $k/3$ times).
  1. They compute $a = u + v$ and $b = x + a$, then $F_0 = f(a)$ and $F_1 = f(b)$. Obviously, this will leak x even if this strategy is used once, which is exactly what the prover wants to prevent.
  2. They compute one of F_0 and F_1 via distributed gate-by-gate computing, and the other by directly computing (e.g., first $u + v = u_B + u_C + v$ and then $f(u + v)$). In this case, even if the protocol will not time-out, the adversary can succeed in convincing the verifier with probability at most $2^{-(k/3)}$ while preserving the secrecy of the secret key x, if this is the majority strategy.
  3. They compute both F_0 and F_1 via distributed gate-by-gate computing. In this case, the timing assumption $\alpha > \gamma+\delta-\tau$ implies that the cheating provers can not present both F_0 and F_1 before the protocol time-out in any iteration.

Proof-of-knowledge. Using standard arguments, the knowledge extractor asks both questions on an iteration; if the prover has better chance than 2^{-k} then in a single iteration it needs to be able to answer both challenges $c \in \{0, 1\}$, which gives away the secret key.

Zero-knowledge. We can construct a simulator to obtain a distribution perfectly indistinguishable from the verifier's view in a real runtime. The simulator guesses the verifier's challenge c. If $c = 0$, the simulator chooses $d, v \in_R Z_q$, computes $F_0 = f(d)$ and $U = g^{d-v} \ mod \ p$, chooses at random F_1 from the range of f. If $c = 1$, the simulator chooses $d, v \in_R Z_q$, computes $F_1 = f(d)$ and $U = g^{d-v} \cdot y^{-1} \ mod \ p$, chooses at random F_0 from the range of f. The simulation can compute within expected polynomial time. □

4 A More Powerful Attack

We now consider a stronger adversary: a new class of "combined software engineering and cryptography" adversarial capabilities, where the adversary is able to break the potential solutions proposed in [JSI96] as well as the timing assumption mentioned above. In the new model, the attackers (say, Bob and Cindy) employ both software (as well as hardware) engineering (read: hacking) and cryptography. Therefore, they can cooperatively *code and debug* some software module (e.g., for cryptographic functions) that is thus trusted by them. (We assume, of course, that the system used by Bob and Cindy to develop the software module is trusted by them. Since Cindy is smart enough not to be fooled, Bob cannot cheat Cindy into using, for example, a malicious compiler as shown in [T84].) The module may be further transplanted into a hardware device (e.g., a smart card). As a result, Bob and Cindy can combine or split keys before or after the process for applying a "not shared" certificate. These attacks suggest the creation of a modified "end-to-end" environment where trust relationships are better handled and better shielded.

4.1 Breaking Faraday Cage-Based Solution

The module (alternatively, the smart card embedded with the module) is intended to accept the dishonest provers' secret shares and then combine them together before the proving process for applying a "not shared" certificate (as shown in Figure 2). A possible attack scenario could be the following:

1. Bob and Cindy distributively generate Bob's secret key (e.g., $x = x_B + x_C$).
2. They cooperatively generate (i.e., code and debug) a software module which is thereafter trusted by them (e.g., there are no Trojan Horses to leak the intended secret).
3. The trusted module is transplanted into a smart card.
4. The smart card accepts their private shares (i.e., x_B and x_C, respectively) and combines them together to obtain Bob's secret key via $x = x_B + x_C$. Therefore, it is the *trusted* software module or smart card that knows Bob's secret key, but not Bob himself.

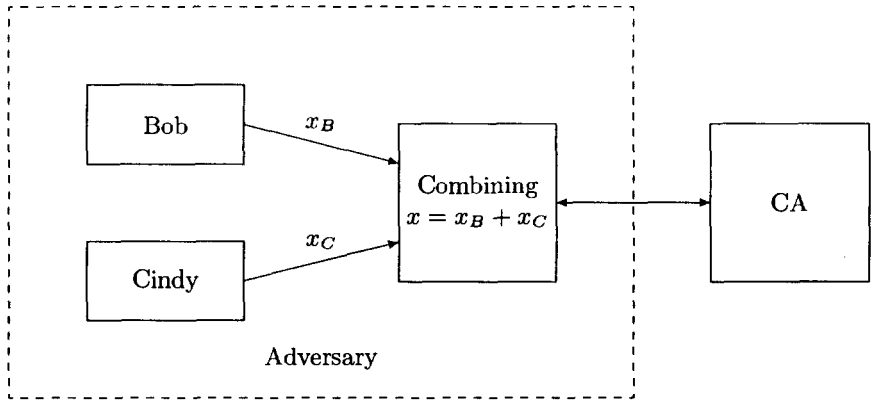


Fig. 2. Breaking Faraday cage-based solution

5. The smart card applies a "not shared" certificate on Bob's behalf.
6. Finally, the smart card is destroyed (e.g., the software module will automatically destroy the secrets) or is kept in a safe such that neither Bob alone nor Cindy alone can open it.

The attack is possible since the existence of a Faraday cage does not necessarily mean that the module (alternatively, the smart card equipped with the module) can not communicate with Bob and Cindy *before* the commencement of the verification process. Also, the cage (e.g., an isolated device) trusted by the verifier does not necessarily imply that the smart card can automatically be trusted by the verifier.

4.2 Breaking Perfectly Trusted CA-Based Solution

It is not hard to see that if Bob's secret key is sent to the CA rather than being generated by the CA, the above attack is possible. Now, consider a more radical strategy and let the CA generate the users' secret keys. Unfortunately, a system under such a strong trust model is still not safe (as depicted in Figure 3), due to potential trusted interactions after the verification stage:

1. Bob and Cindy cooperatively generate (i.e., code and debug) a software module which is thereafter trusted by them (e.g., there are no Trojan Horses to leak the intended secret).
2. The trusted module is transplanted into a smart card.
3. The smart card accepts Bob's secret key and the corresponding certificate from the CA.
4. The smart card shares Bob's secret key via an appropriate secret sharing scheme (e.g., $x = x_B + x_C$). Again, it is the *trusted* software module or smart card that knows Bob's secret key, but not Bob himself.

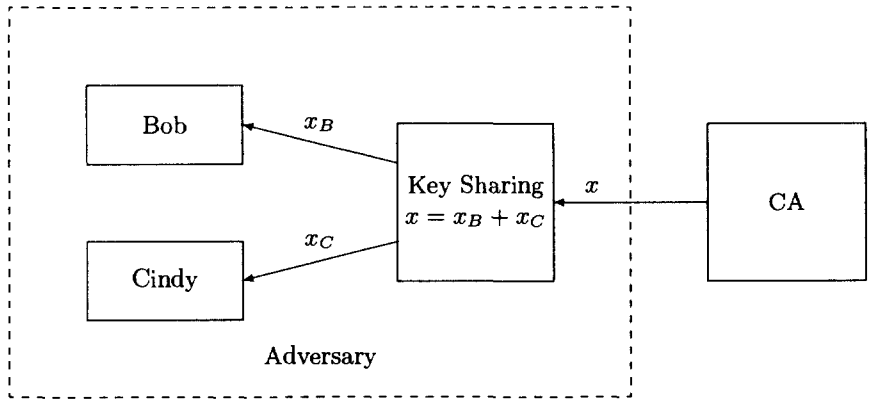


Fig. 3. Breaking trusted CA-based solution

5. The smart card sends Bob and Cindy their shares of "Bob's secret key" (i.e., x_B and x_C respectively) and the "not shared" certificate is publicly available.
6. Finally, the smart card is destroyed (e.g., the software module will automatically destroy the secrets) or kept in a safe such that anyone of them cannot open it.

5 A New Solution

In order to deal with the "combined software engineering and cryptography" attack, we present a new solution based on tamper-resistant components (similar to observers) which are given by the CA to the respective users after certain processing. Due to the introduction of the tamper-resistant components (typically, smart cards), the model has to be refined. As we will see, this is necessary (including the analysis of trust relationship implied by the system where various hardware/software components are provided by different parties).

In Section 5.1-5.4, we focus on simple applications like "designated verifier proofs" [JSI96], where a smart card is just used to check whether or not the received proofs are valid. The model and analysis can be seamlessly and modularly incorporated into more advanced applications like [GJM99], where a smart card is also used to generate certain messages (e.g., private contract signatures) which are then forwarded by the card holder to the outside world. In Section 5.5, we present a case study for generating private contract signatures [GJM99].

5.1 The Model

The Participants. There are three categories of explicit participants: the users who apply for "not shared" certificates, the smart cards, and the CA. There is also an implicit participant, the smart card provider (who plays no malicious

role in our trust model specified below). The CA gets smart cards from the chosen smart card providers, and gives them to individual users after appropriate processing (e.g., initializing some cryptographic parameters and installing some software programs). As said before, the CA has the incentive to do this because issuing "not shared" certificates is a value-added service. The users then apply for "not shared" certificates by presenting certain "proofs" that are generated by the software program installed in their smart cards.

The Trust. In order to simplify the system, we claim the following trust relationship.

- The smart card hardware is tamper-resistant in the sense that it will erase the secrets stored in it if there is an attempt at breaking into it.
- The smart card software (e.g., operating system) is secure in the following sense: (1) it will not tamper with any application software program installed in it; (2) there is no back-door for leaking the secrets stored in the smart card; (3) it provides a secure pseudorandom generator, if necessary.
- The CA will not collude with any user to issue him or her a "not shared" certificate without a correct "proof" generated by the corresponding smart card. This also implies that the software program installed in the smart card by the CA (for generating temporary certificates) will not leak any information about the private keys to the card holders.
- The users' computers (including the software programs) are secure. In particular, the randomness output by the software program is guaranteed to be chosen uniformly at random, and there are no Trojan Horses in their computers.

The Communication. Once a user, Bob, has obtained his smart card from the CA, the smart card cannot communicate with the outside world except the corresponding holder, Bob. We assume that the communication channel between Bob and his smart card is physically secure. However, all the other communication channels (including the one between Bob and the CA) are public and thus subject to eavesdropping by any party.

The Adversary. We assume that Bob will not let Cindy have complete control over his private key x (i.e., self-enforcement). Given the above trust model, we consider the following potential attacks.

- The dishonest Bob manages to obtain a "not shared" certificate for his public key g^x corresponding to the private key x that is cooperatively generated by Cindy and himself.
- The dishonest Bob manages to share the private key x corresponding to the public key g^x that is generated by the software program installed by the CA in his smart card and thus certified by the CA via a "not shared" certificate.
- Although the CA is trusted to behave honestly in issuing "not shared" certificates, it may not be trusted in any other sense. For example, the software

program installed by the CA for generating "proofs" may not be trustworthy and may intend to leak information via a subliminal channel [S98] or a kleptographic channel [YY97]. As a result, the private key may be completely compromised after further cryptanalysis. For simplicity, we assume that the software program will not adopt the *halting* strategy [D96] to construct a subliminal channel for leaking secret information to the CA. By *halting* strategy we mean that the software program installed by the CA decides whether or not to generate a valid message that is requested by the smart card holder. For example, the software program does generate the requested message only when it has embedded certain information into the subliminal channel known to the CA. We also assume that the software program will not adopt the *delaying* strategy to construct a subliminal channel. By *delaying* strategy we mean that the software program outputs a valid response to a request from the smart card holder only at the time that coincides with a predefined subliminal time channel. For example, a message generated in the morning meaning the bit of 0, and a message generated in the afternoon meaning the bit of 1. In short, we assume no covert channels.

5.2 The Objective

The objective of the protocol is to ensure that one's private key is known to himself (i.e., secret non-sharing), while no information about the private key is leaked to any participant. Since we will adopt a signature scheme for the software program to generate temporary certificates for the users' public keys, we must guarantee that this process leaks no information about the secrets of the users. For this purpose, we use a subliminal-free signature system, where subliminal-freeness is in the sense that the signer (i.e., the software program installed in a smart card by the CA) cannot abuse the randomness for generating signatures to establish a subliminal channel.

Definition 2. *Suppose the software program installed by the CA in a smart card will output a valid signature within a reasonable delay of computation whenever it is invoked by the card holder. The resulting signature system is subliminal-free, if:*

- *The time at which the software program generates a temporary certificate is completely under the card holder's control.*
- *The to-be-signed messages are guaranteed to be distributed uniformly at random in the corresponding message space.*
- *The randomness used to generate signatures is guaranteed to be chosen uniformly at random. (In the case that the signature generation algorithm is deterministic like the "full-domain hash and RSA"-based signature scheme, this is naturally satisfied.)*

Now we are ready to define the security for a smart card-based solution to ensuring *secret non-sharing.*

Definition 3. *A smart card-based solution to ensuring secret non-sharing is secure, if the followings are satisfied simultaneously.*

- *No dishonest user can obtain a "not shared" certificate for the public key g^x corresponding to a private key x that is cooperatively generated by him and a third party.*
- *No dishonest user can share the private key x corresponding to the public key g^x that is generated by the software program installed by the CA and thus certified by the CA via a "not shared" certificate.*
- *The signature system whereby the software program generates temporary certificates for the users' public keys is subliminal-free.*

5.3 The Solution

In this solution, each participant (e.g., Bob) applying for a "not shared" certificate needs to obtain a tamper-resistant smart card from the CA. Each card is equipped with a software program SP provided by the CA, a native temporary pair of public and private keys (pk_{SP}, sk_{SP}), a pair of generators g and h of order q in the subgroup G_q of Z_p^* that is publicly verifiable as in the standard discrete logarithm setting, and a value $g^a h^b$ such that $a, b \in Z_q$. We stress that the discrete logarithm $\log_g h \ mod\ q$ is known to the software program (and possibly also known to the CA) but not to any other participant. When Bob applies for a "not shared" certificate, he interacts with the software program SP installed in the smart card by the CA to generate a private key x that is unknown to Bob and never departs the card, and to obtain the SP's signature for his public key g^x. Then, Bob presents this signature (i.e., a temporary certificate) to the CA to exchange a "not shared" certificate for his public key g^x. In order to make concrete our solution and to clarify the security of the resulting system, we assume that the software program SP adopts the Schnorr signature scheme [S91] to sign temporary certificates. The protocol whereby Bob applies for a "not shared" certificate goes as follows:

1. The software program SP sends the equipped $g^a h^b$ as well as its temporary public key $pk_{SP} = g^{sk_{SP}}$ to Bob. If necessary, the software program also sends g, h, p, q to Bob who can thus verify if both g and h are of order q.
2. Bob chooses $z \in_R Z_q$ and sends z to the software program SP. Now Bob can himself compute his public key as $pk'_{Bob}=(g^a h^b)/h^z$.
3. The software program SP calculates the private key for Bob as follows: $x = a + (b - z) \cdot \log_g h \ mod\ q$. Denote $pk_{Bob}=g^x \ mod\ p$. Then, the software program chooses $r \in Z_q$, and sends $R = g^r \ mod\ p$ to Bob. Note that r may not be chosen uniformly at random.
4. Bob chooses $r' \in_R Z_q$ uniformly at random and sends it to the software program SP.
5. The software program SP generates a Schnorr signature for Bob's public key $pk_{Bob} = g^x$ as follows. It computes $r^* = r + r' \ mod\ q$, $c = H(pk_{Bob}, g^{r^*})$, and $d = c \cdot sk_{SP} + r^* \ mod\ q$, where H is an appropriate hash function (i.e., what we idealized as an instantiation of a random oracle) with range Z_q. Finally, it sends the signature $SIG_{SP}(pk_{Bob}) \stackrel{def}{=} (pk_{Bob}; c, d)$ to Bob.
6. Bob checks if $pk_{Bob} \stackrel{?}{=} pk'_{Bob}$, $g^d \cdot (pk_{Bob})^{-c} \stackrel{?}{=} R \cdot g^{r'}$, and $c \stackrel{?}{=} H(pk_{Bob}, R \cdot g^{r'})$. If all the verifications pass, Bob sends $SIG_{SP}(pk_{Bob})$ to the CA to exchange

a "not shared" certificate $Cert_{CA}(pk_{Bob})$; otherwise, the software program is cheating and the CA is disqualified.

Remark 4. If we adopt an even stronger trust model in which the CA is completely trusted, which means that any software program installed by the CA behaves honestly, then the solution could be simplified since we need not to worry about the existence of any subliminal channel.

Remark 5. The software program must commit to various parameters first; otherwise, there could be other subliminal channels. For example, it has two pairs of public and private keys and it chooses which one to use based on certain bit of the user's private key.

Remark 6. Self-enforcement implies that Bob will not let Cindy have complete control over his smart card. On the other hand, it is unrealistic for Cindy to participate in all the processes (i.e., from obtaining Bob's smart card to verifying a proof designated for Bob) rather then just the process for verifying a designated verifier proof.

5.4 Security Analysis

Proposition 1. *Suppose the software program SP installed by the CA in the smart card will output a valid signature within a reasonable delay whenever it is invoked by the card holder. Then, the resulting signature system is subliminal-free.*

Proof (sketch) By assumption, the smart card will not adopt the *halting* strategy to establish a subliminal channel.

- The time at which the software program generates temporary certificates is completely under the control of the corresponding card holder, since we assume that the software program responds to a request from the user within a reasonable (more or less expected) delay.
- The to-be-signed messages are guaranteed to be chosen uniformly at random in the corresponding message space. This is since no matter how the software program chooses g, h, $g^a h^b$, and (pk_{SP}, sk_{SP}), Bob's private key x is uniformly distributed in Z_q since z is uniformly chosen at random, where $x + z \cdot log_g h = a + b \cdot log_g h$. Therefore, the corresponding public key g^x is also uniformly distributed in G_q.
- The randomness used to generate signatures is guaranteed to be chosen uniformly at random. This is since no matter how the software program chooses r, $r^* = r + r'$ is uniformly distributed in Z_q since r' is uniformly chosen at random in Z_q. Thus, g^{r^*} is also uniformly distributed in G_q. □

Theorem 2. *The smart card-based solution of ensuring secret non-sharing is secure.*

Proof (sketch) We argue that the requirements in Definition 3 are satisfied.

- No dishonest user can obtain a "not shared" certificate for his public key g^x corresponding to the secret x that is cooperatively generated by him and Cindy. This is due to the existential unforgeability of Schnorr signature scheme in the random oracle model, which is used by the software program to generate temporary certificates.
- No dishonest user can share the private key x corresponding to the public key g^x generated by the software program and thus certified by the CA via a "not shared" certificate. This proof is done in two steps.

  First, suppose Bob can obtain the corresponding private key x with non-negligible probability, then we can break the discrete logarithm assumption as follows. Suppose we are given $h \in_R G_q$ generated by g and intend to calculate the discrete logarithm $log_g h$. After installing the software program in the smart card, the challenge h is given to the software program that is equipped with a pair of public and private keys (pk_{SP}, sk_{SP}).

  1. The software program chooses $a, b \in_R Z_q$ and sends $g^a h^b$ to Bob.
  2. Bob chooses and sends $z \in Z_q$ to the smart card. Note that z may not be chosen uniformly at random, but this does not matter.
  3. The software program calculates $pk_{Bob} \stackrel{def}{=} g^x = g^a h^b / h^z \ mod\ p$. Then, the software program chooses $r \in_R Z_q$, and sends $R = g^r \ mod\ p$ to Bob.
  4. Bob chooses $r' \in Z_q$ and sends it to the software program.
  5. The software program computes $r^* = r + r' \ mod\ q$, $c = H(pk_{Bob}, g^{r^*})$, and $d = c \cdot sk_{SP} + r^* \ mod\ q$, where H is an appropriate hash function with range Z_q. Finally, the software program sends the signature $SIG_{SP}(g^x) \stackrel{def}{=} (g^x; c, d)$ to Bob.
  6. The signature $(g^x; c, d)$ is always valid, since the software program knows sk_{SP}.

  Therefore, if Bob can obtain the private key x with non-negligible probability, then the discrete logarithm assumption is broken since the software program can calculate $log_g h$ from (a, b) and (x, z).

  Second, we guarantee that if Bob tells Cindy that he has a way to share the private key x corresponding to the public g^x that was certified by the software program as well as by the CA, then Bob really knows the private key x. No matter what algorithm and software they use, finally Bob gets his share x_{Bob} and Cindy gets her share x_{Cindy} with the supposed property (for example) that $x = x_{Bob} + x_{Cindy}$, whereas $g^{x_{Bob}}$ and $g^{x_{Cindy}}$ are known to each other such that $g^x = g^{x_{Bob}} \cdot g^{x_{Cindy}}$. If Bob succeeds in cheating Cindy in the sense that Bob does not know the discrete logarithm of $(g^x)^{-x_{Cindy}}$ with respect to base g, it will be an issue associated with Bob. This is so since Cindy is smart and she asks Bob to prove in zero-knowledge that he knows x_{Bob} corresponding to $(g^x)^{-x_{Cindy}}$. Therefore, if Bob can share x with Cindy, without each one of them knowing x corresponding to the certified public key g^x, the knowledge extractor will guarantee that x_{Bob} can be known. That is, x can be known to Bob (whose implication was shown above).

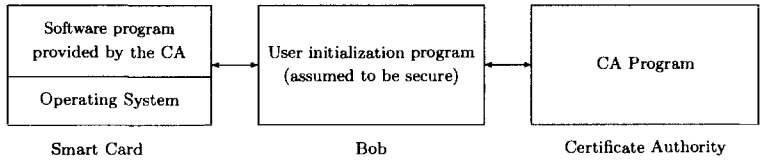
– The signature system whereby the software program SP generates temporary certificates for the users' public keys is subliminal-free (by Proposition 1). □

5.5 Secure Private Contract Signatures: A Case Study for Advanced Applications

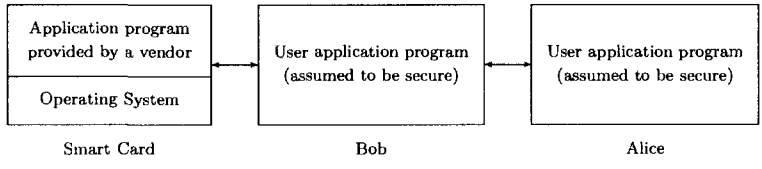
In the above we focused on simple applications where smart cards do not send any information to the outside world after initialization. However, for those applications like [GJM99] where a smart card needs to generate *private contract signatures* that will be sent to Bob's business partner, say Alice, over a public channel, there is the possibility that the application program AP provided by an application software vendor may leak information about Bob's private key via a subliminal channel. The architecture for such advanced applications is depicted in Figure 4, where the upper half also corresponds to applications like *designated verifier proofs.*

Although the application program AP for generating *private contract signatures* may not be trusted, we still assume for simplicity that: (1) it will not leak any information about the private keys to the card holders; (2) it will not adopt the *halting* strategy to establish a subliminal (covert) channel; and (3) it will not adopt the *delaying* strategy to establish a subliminal channel. The rationale for these assumptions can be justified by the fact that the application software provider might not take the chance of establishing subliminal and covert channels due to the fear of losing its revenue (assuming the software is scrutinized).

Private Contract Signatures. Before we define what is a secure private contract signature system, we briefly review what is a private contract signature [GJM99]. Say Alice has public key y_A and private key x_A, where $y_A = g^{x_A}$. Similarly, Bob has a pair of public and private keys (y_B, x_B), and T has a pair of



(a). The interactions in applying a "not shared" certificate



(b). The interactions in generating "private contract signatures"

Fig. 4. The architecture of a "private contract signature" system

public and private keys (y_T, x_T). In order for Alice to generate a private contract signature on m for Bob, she sends Bob a proof of the statement

"X is a T-encryption of '1' AND I can sign m as Alice
OR
X is a T-encryption of '2' and I can sign m as Bob"

where X is some value, and T-encryption denotes a message encrypted with T's public key. Alice can do this because she can perform a T-encryption of '1' to generate X, and she can sign m herself. After exchanging the private contract signatures in the trustee-invisible scheme, Alice sends to Bob a proof of the statement (we call it "proof of decryption," done by Alice or T)

"Alice: X is a T-encryption of '1' OR T: X is a T-encryption of '1'".

Definition 4. *Suppose the application program in the smart card will output a valid response within a reasonable delay whenever it is invoked by the card holder. A private contract signature generation system is subliminal-free, if:*

- *The time at which the application program generates the private contract signatures is completely under the user's control.*
- *The ciphertext X and the message m are completely under the user's control.*
- *The randomness used to generate private contract signatures and proofs of decryptions is guaranteed to be chosen uniformly at random.*

Now we are ready to define what it means for a smart card-based private contract signature system to ensure secret non-sharing. Note that this requirement does not exist in the context of [GJM99] where the private key is assumed to be known to Bob, while security properties of private contract signature systems proved there should be inherited into the extended private contract signature system we will specify. Since those properties are naturally inherited, we only consider the security requirements inspired by the incorporation of smart cards.

Definition 5. *A smart card-based private contract signature system ensures secret non-sharing, if the followings are satisfied simultaneously.*

- *The process of issuing a "not shared" certificate is secure (see Definition 3).*
- *No dishonest user can share the private key x corresponding to the public key g^x generated by the software program installed by the CA and thus certified by the CA via a "not shared" certificate, even after obtaining polynomially many private contract signatures as well as the corresponding "proofs of decryptions" from the application software in the smart card.*
- *The private contract signature generation system is subliminal-free.*

The Private Contract Signature Generation System. Suppose Bob is to generate a private contract signature on message m for Alice. Bob interacts with the application program AP in his smart card via the following protocol.

1. Bob generates the T-encryption of '2' and then sends the randomness used in generating the encryption to the application program.
2. The commitment for fulfilling "I can sign m as Bob" is computed as follows: the application program chooses and sends $R = g^r$ to Bob; Bob chooses and sends $r' \in_R Z_q$ to the application program; the commitment is defined as $R \cdot g^{r'}$.
3. The application program and Bob collaboratively choose a triple $(c, d_1, d_2) \in_R Z_q^3$. This can be easily done by assuring the application program commits to its randomness first.
4. The application program can generate (via deterministic algorithms) a private contract signature in which (c, d_1) and (c, d_2) are used to simulate the proof that "X is a T-encryption of '1'" and "I can sign m as Alice", respectively. Moreover, the application program can generate (via deterministic algorithms) (c', d_1') and (c', d_2') for the proof that "X is a T-encryption of '2'" and "I can sign m as Bob", respectively.
5. In order to generate the "proofs of the decryptions", Bob and the application program collaboratively generate $(c_2^*, d_2^*) \in_R Zq$ for the proof "T: X is a T-encryption of '2'", then the application program can generate (via deterministic algorithms) (c_1^*, d_1^*) for the proof "Bob: X is a T-encryption of '2'". This can be easily done by assuring the application program commits to its randomness first.

Security Analysis. Similar to Proposition 1, we have

Proposition 2. *Suppose the application program in the smart card will output a valid response within a reasonable delay whenever it is invoked by the card holder. Then, the above private contract signature generation system is subliminal-free.*

Now we are ready to prove that the private contract signature system is secure.

Theorem 3. *The above smart card-based private contract signature generation system ensures secret non-sharing.*

Proof (sketch) We prove that the above system satisfies Definition 5.

- The process of issuing "not shared" certificates is secure. This is exactly the same as in Theorem 2, since it is just a module that is seamlessly integrated into the private contract signature systems.
- Even after polynomially many queries to the application program for generating private contract signatures as well as proofs of decryptions, no dishonest user can share the private key x corresponding to the public key g^x that is generated by the software program installed by the CA and thus certified by the CA via a "not shared" certificate. The proof also has two steps.

First, suppose Bob or Cindy can obtain the private key with non-negligible probability, then we can break the discrete logarithm assumption. Basically, we can simulate the real-world process for generating private contract signatures by letting the application program know Alice's private key, and the real-world process for generating proofs of decryptions by letting the application program know T's private key. Second, if Bob can succeed in convincing Cindy via a zero-knowledge proof of knowledge that their shares are correct with respect to the private key x, the knowledge extractor guarantees that x can be calculated. We omit the details.

- The private contract signature generation system is subliminal-free. This is proved in proposition 2. □

6 Conclusion

We proposed a cryptography-based timing assumption to replace the physical assumption of the existence of a Faraday cage, based on which we also presented a new notion we call Verifiable Secret Non-Sharing. This solves the problem of secret non-sharing where the parties do not use preprocessing in putting their shares together in order to cheat in the verification.

Next, we introduced a class of "combined software engineering and cryptography" attack that is able to corrupt all of the previously proposed solutions. We constructed a tamper-resistant hardware-based solution where the certificate holders are forced not to share their secret keys with any other party. We postulated a concrete model of the system and claimed properties it achieves. It remains as an interesting area of research to formalize the model and the various assumptions we make as a concise model for secure trust model for computing with trusted elements and tamper-proof devices. This is much needed, since the area of trusted computing in the presence of potential leakages is lacking rigorous treatment.

Acknowledgment. We thank the anonymous reviewers for useful comments. Shouhuai Xu was partially supported by an NSF grant to Laboratory for Information Security Technology at George Mason University.

References

[BBD+91] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, and J. J. Quisquater, Secure Implementation of Identification Systems, Journal of Cryptology, 1991 (4), pp 175-183.

[BC93] S. Brands and D. Chaum, Distance-Bounding Protocols, Eurocrypt'93.

[BCC88] G. Brassard, D. Chaum, and C. Crepeau, Minimum Disclosure Proofs of Knowledge, Journal of Computer and System Science, Vol. 37, No. 2, Oct. 1988, pp. 156-189.

[BD90] T. Beth and Y. Desmedt, Identification Tokens-or: Solving the Chess Grandmaster Problem, Crypto'90.

[BN00] D. Boneh and M. Naor, Timed Commitments, Crypto'00.

[BR93] M. Bellare and P. Rogaway, Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols, ACM CCS'93.

[CvA89] D. Chaum and H. van Antwerpen, Undeniable Signatures, Crypto'89.

[D96] Y. Desmedt, Simmons' Protocol Is Not Free of Subliminal Channels, Computer Security Foundation Workshop'96.

[DF89] Y. Desmedt and Y. Frankel, Threshold Cryptosystems, Crypto'89.

[DF91] Y. Desmedt and Y. Frankel, Shared Generation of Authenticators and Signatures, Crypto'91.

[DNS98] C. Dwork, M. Naor, and A. Sahai, Concurrent Zero-Knowledge, STOC'98.

[DY91] Y. Desmedt and M. Yung, A Weakness of Undeniable Signature, Eurocrypt'91.

[G98] O. Goldreich, Secure Multi-Party Computation, 1998.

[GJM99] J. Garay, M. Jakobsson, and P. MacKenzie, Abuse-Free Optimistic Contract Signing, Crypto'99.

[GM99] J. Garay and P. MacKenzie, Abuse-free Multi-party Contract Signing, DISC '99.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, How to Play any Mental Game-A Completeness Theorem for Protocol with Honest Majority, STOC'87.

[J94] M. Jakobsson, Blackmailing Using Undeniable Signatures, Eurocrypt'94.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo, Designated Verifier Proofs and Their Applications, Eurocrypt'96.

[K96] P. Kocher, Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS, and Other Systems, Crypto'96.

[KR97] H. Krawczyk and T. Rabin, Chameleon Hashing and Signatures, NDSS'2000.

[S79] A. Shamir, How to Share a Secret, CACM, Vol. 22, No. 11, pp 612-613, 1979.

[S91] C. P. Schnorr, Efficient Signatures Generation by Smart Card, J. of Cryptology, 4(3), 1991, 161-174.

[S96] M. Stadler, Publicly Verifiable Secret Sharing, Eurocrypt'96.

[S98] G. J. Simmons, The History of Subliminal Channels, IEEE Journal on Selected Areas in Communication, vol. 16, no. 4, May 1998.

[T84] K. Thompson, Reflections on Trusting Trust, CACM, Vol. 27, No. 8, pp 761-763.

[Y82] A. Yao, Protocols for Secure Computations (extended abstract), FOCS'82.

[YY97] A. Young and M. Yung, Kleptography: using Cryptography Against Cryptography, Crypto'97.

Split-and-Delegate: Threshold Cryptography for the Masses

Daniel E. Geer, Jr.[1] and Moti Yung[2]

[1] @stake Inc.
geer@atstake.com.
[2] CertCo Inc.
moti@cs.columbia.edu.

Abstract. Threshold Cryptography (distributed key splitting) is traditionally employed as means to preserve the whole key against compromise, i.e., for risk reduction (coping with memory compromise) and availability (coping with denial of service). Recently, some functionality of splitting keys has been shown to be useful beyond preservation, yielding a small number of high-security, server-related applications. However, the business applications and market applicability of splitting keys is still not realized or analyzed. The goal of this *position paper* is to put forth the thesis that the full power of threshold cryptography as a useful and attractive tool is going to be unleashed only if the ability to split keys is given to end users (the masses). More specifically, we claim that threshold cryptographic operations (e.g. splitting a key) together with user capability to delegate (which we view as a necessary extension of PKI) should be part of the suite of operations available to end-users of a PKI (e.g., embedded in user crypto-APIs / user smartcards). This new tool ("split and delegate") will enable flexible key management at the user level, in contrast with the traditional rigidity of PKI. We note that threshold cryptography is currently mainly an idea and still not in the market (though some companies do offer split key in hardware or software). We believe that the economic value of the suggested user-based applications will be the central driving force behind any market adoption of threshold cryptography. We give an analysis of the potential business and of market penetration scenarios (such business analysis of suggested new cryptographic applications is often done but rarely published).

Keywords: PKI, applications, enabling technology, user Crypto-API, smartcard, threshold cryptography, splitting keys, delegation, business analysis, market penetration.

1 Introduction

All available security technology manages the relation between some class of secrets and some class of truth. It is the mapping between secrets and truth that determines all other characteristics of the particular regime. Public Key

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 220–237, 2003.

Infrastructure (PKI) is the generic catch-all phrase for the apparatus of security by way of public key technology. This public key technology creates an asymmetry of keys: a private key half and its corresponding public key half. PKI, in fact, is about managing the promiscuous distribution of the public half of keys and the chaste protection of the private halves. Virtually all PKI's costs are incurred during verification of the validity (un-revoked status) of a key already in circulation.

Threshold cryptography takes this notion of key halves further and splits the private key-half into fragments and distributes those fragments amongst different holders. Its first refinement, quorumed fragmentation, mitigates the costs of high availability for the fragments as only m-of-n fragments are collectively necessary to recreate the original key. One's cost-benefit-risk tradeoff is between complexified key (fragment) management and the complexified barriers a defrauder must overcome. Putting it differently, quorumed key fragmentation is at once the most powerful current design weapon against both single actor fraud and single actor denial of service.

Or so the story has been told. As always, security technology is cost-dominated whenever it is construed as a disabling (protective) technology, by which we mean a technology whose principal contribution is that of disabling something (an attack) or someone (an attacker). A disabling technology returns nothing of value, per se. Rather it is a tax on productivity, hard to justify as an ROI (return on investment) proposition, and likely to be resisted unless the tax can be finely subdivided such as in a transaction oriented environment. When a technology can only be described as a disabling technology, even the best example of it will rarely find a market. In the case of quorumed key fragmentation, it has (had) no apparent use except for the most principled certifying authorities (CAs) [6] and servers of similar sensitivity (escrow agency [6], internal CA organization [15], tallying authority [4], proxy signing or encryption server [2, 18]) – in total a seemingly tiny market By contrast, an enthusiastic demand pull requires a focus on enablement, not disablement – a demonstrated reason to buy, a positive ROI that does not depend on avoiding situations you would not want even to simulate. Indeed, key fragmentation has been recently added as a capability of a hardware or a software module by some companies, though still for the purpose of disabling advanced attacks. The question, then, is whether it can be made a centrum for an enabling technology and thus be attractive in the business world.

Cryptographic technology is made available within some systems (based on some initial belief of its applicability) and is not made available with other systems (based on a some initial belief in its inability to contribute). Our goal in investigating threshold cryptography as an enabling technology is to look at the business case of embedding this technology within various APIs and platforms, a focus too often ignored when cryptographic technologies are first implemented. Exactly where threshold cryptography is made available has major implications on its applicability since the availability and durability of code within an architecture, and to which entities it is given, is crucial and lasting as is argued in [20].

We posit that key fragmentation has a wealth of application outside of highly protected services such as root CAs, but this obtains if and only if the fragments can be arranged to not complexify key management but merely extend the applicability of cryptographically sophisticated means. This paradoxical result depends on overturning the conventional formulation of a public key hierarchy such that many costs of a general purpose PKI can be avoided, especially those that are due to the rigors of stranger-to-stranger introduction by third parties and the complexity costs of ensuring that all consumers of credentials are equally informed of revoked identities. This overturning may be essential in any case as, perversely, the procedural complexity of absolute key validation (by way of revocation testing) rises as the square of the distance from the root the subject key is positioned even though the further the key is from the root the less important its protection is likely to be as a practical matter. This is a complexity cost that runs in precisely the wrong direction and it will defeat economic application of public key cryptography as it is presently defined. PKI components are not now, and we argue cannot become, a proletarian commonplace unless these complexity costs are mitigated.

We begin with an idea on that narrow point. A certificate is a file format in which to bind a verifier to an assertion, that is, the public key half to the so-called "DN" or "Distinguished Name" (along with various administrative details we set aside for the moment). A certificate declares that, at such and such a time, the Issuer committed itself to the binding by jointly signing the DN and the key half (at least) with the Issuer's own (private) key.

The X.509 standard posits a hierarchy of certifying authorities, each issuing key-to-DN bindings and each, in turn, certified by its upstream CA, excepting the "root CA" whose certificate, like Napolean's sovereignty, is self proclaimed. Yet, there is nothing inherently hierarchic about certificate issuance; hierarchy is only an issue in verification, both as to a chain of signatures upward to a root and as a source of evidence that those signatures are made in keys as yet un-revoked. It reflects the hierarchic interplay of power and of risk. A certificate itself can, in principle, be created by anyone; it is only virginal thinking that has lead us to hierarchic PKIs. It is only our lack of imagination that has made personal identity checks relative to some certifying authority the bulk of PKI's design focus and of certificate traffic. In fact, the PGP approach [16] and the SPKI/SDSI [11] approaches to PKI allow everyone, in some sense, to be a CA.

So, relax your presumptions for a few minutes. Put a CA in your wallet. Run it on a smartcard, a PDA, or something hybrid. Generate keys and split them. Issue certificates as you will. Delegate to others what you would have them do for you (i.e., split-and-delegate). Make the everyday use of PK technology one that begins with a personal CA and flows outward by having other authorities accept and endorse what the individual has emitted rather than have remote authorities issue you certificates or, worse, issue both the certificates and the key pairs. Invert the hierarchy, give this power to the masses.

2 Applications of Key Splitting

"Threshold cryptography" is a highly developed topic, mainly for discrete-log based systems and RSA, (in fact, we cannot and do not attempt to cover the entire body of important cryptographic papers herein). The methodology was started in [7,3,12,8,9] Provably secure schemes were then designed; see [6,17]. Then adding adversary capabilities was considered in various works. Proactive systems [22] (and schemes in, e.g. [17,13]) allow the fragments to be renewed and rerandomized which protects even against a mobile adversary that over time compromises many systems' components. It also allows for the changing of the set holding fragments and resplit of keys [22,14,10].

Given the above systems, let us view the technology at a very high level of functionality which is sufficient for application development. The characteristics of any key splitting technology we favor are (1) that a key can be split (n-of-n or k-of-n) and, if needful, dynamically re-split, (2) that fragmentation can quorumed (i.e., only m-of-n fragments are needed, and any m will do), (3) that sign/seal operations can be performed step-wise such that asynchronously and partially completed operations do not leak information, (4) that one fragment of a multiway split can be fixed by design, (5) that a security kernel (cryptographic device) can refuse to leak key fragments once gotten, and (6) that operations can be assured to be correct (due to robustness) in case parties are not trusted.

For completeness, we give an example (based on known techniques) and recall the splitting of an RSA key into two pieces. An RSA private key is an exponent d which is to be used over a (hashed) message M and produce a signature $S = M^d \bmod N$ (N a composite which is a product of two unknown large primes). Then we can assume that a random element g is given together with its signature: $g^d \bmod N$ (a witness). In this case, the owner of the key O_1 splits its key into two pieces by choosing a random element in $d_1 \in [-N, N]$ and assigning $d_2 = d - d_1$. The owner then sends privately to the second server O_2 the value of d_1 and also publishes in public $g^{d_1} \bmod N$ and $g^{d_2} \bmod N$ (as part of a certificate of the splitting). O_2 can check that the public witness of its share is actually g raised to the share. Also, everyone can check that the two new witnesses correspond to a splitting of the original key by multiplying the two new witnesses and comparing to the old public witness. The above was the split-and-delegate portion of the operation. Then the two parties O_1, O_2 can jointly sign a document. The availability of a witness g, g^{d_i} enables the server which applied d_i to a new message M to prove that indeed the share d_i was used correctly: show that logarithm base g of $g^{d_i} \bmod N$ equals the logarithm base M of $M^{d_i} \bmod N$. This can be done using an efficient zero-knowledge technique, which gives the operation the *robustness* property without compromising the security of this signature scheme.

Note that when we split a key 2-of-2 (n-of-n), each fragment is essential, i.e., if we call the fragments A, B, then a completed signature has embedded in itself proof that both A and B signed or, simply, the satisfaction of the logic "A and B." If we split a key 2-of-3 (based on Shamir's polynomial based sharing, say) and promptly sign with (additional) fragment C (whose partial signature can be added to the final signature), then a completed signature has embedded in

itself proof that either A or B signed or, simply, the satisfaction of the logic "A or B." Ipso facto, we have the two functions of a monotone logic, viz., AND and OR. An n-of-n split provides AND, a 2-of-n followed by a single fragment signature provides OR. Because a given fragment may be further split, we can combine AND and OR conditions as required. In other words, the completion of a cryptographic operation like a signature is, in and of itself, proof that the logic embodied in the key splits was satisfied whatever that may have been. That this is largely asynchronous and fully dispersable represents a new construct, one that is the basis for the claim that we have moved from protective and static to synthetic and dynamic uses of the technology of splitting keys. To claim this result as fundamental, it is instructive to look at some preliminary instances of its application.

We note that for some applications, one can designate an explicit multi-signature operation which recognizes individual keys [3] and an explicit delegate signature scheme [21]. The difference between implicit key split and explicit multi-signing was discussed in [15]. In most of our applications the key splitting approach is most suitable and easy to interface, though in some cases a combination of explicit sharing and an implicit one makes sense.

So, why split keys? We show next a few examples. We start from the already recognized applications and move into new areas where end-users can apply the technology by splitting and delegating keys. Note that we do not claim that our applications can only be achieved by split-and-delegate, rather that splitting keys (due to the flexibility of control it enables at the user level) captures in a single mechanism a vast amount of applicable procedures and processes. Given this fact, the following section suggests market and business analysis based on these applications.

1. **CA:** Loss of a Certifying Authority (CA) key is an expensive failure. Splitting that CA private key half as a quorum and distributing the fragments to independent holders is the strongest risk management known for a CA – no single bad actor then has enough authority to make a fraudulent assertion and no single failed actor is essential enough to derail processing. For a CA of sufficient renown, complexity costs are not germane since nearly any complexity cost will pale in comparison to the (reputation capital) cost of a CA failure subsequent to loss of faith in that private key which is truly the root of the CA's authority. One of the early suggestion of using the technology for a CA appeared in [6].
2. **Revocation:** Just as with a CA, the signature that underscores a revocation assertion must be protected against all attackers and is not really part of an ordinary economic discussion as to means and costs. This is particularly relevant if, as we suppose, a root-key for a revocation hierarchy cannot itself be revoked. (The revocation of a key is always retroactive to a time when the key's protection was known to be good hence a revocation of a signature key repudiates those signatures made since that moment of retroactivity and, further, the assertions so signed likewise lose their validity. Since repudiating a signed revocation arguably reinstates the previously revoked key, this is a confusing path down which none of us will want to go.) Protecting

this un-revocable key with via fragmentation seems almost a tautologous requirement.

3. **Escrow:** Keys are escrowed for two orthogonal reasons, surveillance and data recovery. Surveillance requires that the cryptographic system have a hole in it and an acceptable escrow system would require that that hole be very precisely accounted for were it to be utilized. We leave the crafting of auditable holes to politics and its desserts (where [6] observed that threshold cryptography can be used for distributing the power of the surveillance agency). Data recovery is, by contrast, nearly a universal good and a data recovery system would enable the very substantial crime prevention outcome of making cryptographic filesystems the norm rather than the exception. Split your personal private key into a 2-of-3 quorum, retain one fragment for your smartcard in your wallet, put one fragment in your principal computing device (such as your laptop) and put one fragment in the company vault. If any one of these is lost the other two can cope – this covers loss of your smartcard, loss of your laptop, and your employer's interest in your data should you expire. No one fragment is useful to anyone, so your laptop can run a cryptographic file system and you can still lose your smartcard without further risk either of disclosure or data loss and so forth and so on.
4. **Countersignature:** Split a key into a 2-of-2 quorum and half-sign with one fragment. Keep that one fragment on your smartcard and send the other to your countersignatory encrypted in his public key, then send a partially signed message to your countersignatory without worry as to its integrity inasmuch as integrity loss merely results in a signature that does not verify. Substitute n-of-n should 2-of-2 be insufficient. A proxy server signing using this idea has been recently suggested in [2] – the difference is that the user holds only one fragment throughout, so the revocation of this key is under the complete control of the server.
5. **Voting/ Distributed Decision:** Split a key into a majority rules quorum, say 5 of 9 for a nine person Board of Directors, and any vote is complete when 5 of 9 have signed. Since there is nothing in the resulting signature that reveals whose fragments were involved, this is secret voting per se which we consider a feature. Note that this is "one fragment, one vote," so the fragments can be mapped to shares as easily as heads, etc. If you must have "black-ball" voting, use a separate key for that (or permit individuals to "sign" with a random number thus destroying the ability of the signature to complete). In the latter case, do not enforce a robustness mechanism on individual voters: for a small (constant) number of voting directors, trying all subsets (of, say, 5 of 9) is a doable computational task. We note that using threshold schemes for voting in the context of databases was suggested in [10].
6. **Consensus:** Any n-of-n split is a consensus requirement.
7. **Notarization:** If A and B need to regularly ensure that documents passing between them are notarized, they pick a suitable notary with whom they establish a 3-of-3 quorum. What A issues and B accepts C can notarize as when C performs his operation a checkable signature should result. If C refuses to pass the message forward absent that check, there is no notarization

without the electronic swearing of the two parties and no evidence of failed attempts. Conversely, B can refuse to accept a communication from A unless A has had that communication stamped by notary C before delivery to B, something that B can check using only B's fragment. In other words, who is enforcing for whom is tailorable.

8. **Off-line deferral:** Any signature with an m-of-n quorum in which you participate can be signed by m-1, put aside either in a filesystem or on your smartcard, and made valid at the moment of use by completing the m-of-n quorum. Where m=n, then only you can complete it. Where m<n, any holder of an unexercised fragment can complete. The point is using time delay to your advantage. A server application of a similar notion is the magic-ink signature idea where unblinding of a signature is time controlled by a quorum [19].
9. **Authorization:** Except in the particular case where "ownership" of a DN confers the right of citizenship, i.e., where owning a name in and of itself confers beneficial rights, an identity cert conveys too little information to manage anything; authorization information has to be added. If, however, the newly logged-in user exchanges his identity cert for a similarly structured authorization credential (in practical terms, a role-identity certificate), then ordinary clients (like browsers) could handle them transparently whilst the server side could rely on the client bringing his access control list (ACL) entry with him at request time rather than the server having to look up a certified identity on a local subset of some global ACL. This issued-on-demand authorization certificate would have a short lifetime and would be but partially signed, i.e., it would be signed by the issuer in a fragment of a 2-of-2 quorum one half of which was issued to the subscriber as the culmination of subscription (or n-of-n for a larger group of approvers). Its invalid-as-issued status would permit its unprotected yet safe transit through the Internet and would relieve the authorization certificate server from having to perform any authentication of the client. This scalability advantage is substantial and underlies the successive designs of Kerberos, the Open Software Foundation's Distributed Computing Environment (OSF/DCE) and the Kerberos adaptation found in in Windows NT 5.5, et seq. In Kerberos-speak, the presentation of an identity cert is like a ticket-granting request and the subsequent authorization cert is like a new credential encrypted in the session key shared between the client and the ticket-granting service. In short, the authorization cert can be issued without proof that the client is who he says he is – only if that fact obtains will the client be able to complete the signature and activate the authorization. The implication that completing the signature is an act of liability acceptance should be noted (and might even be encoded in the data covered by the signature thereby establishing the basis for recourse). The split signature, done on the fly, can dynamically control the number and composition of ticket granting approvers without increasing the complexity of the ticket beyond that of a cert.
10. **Expiry bridging:** No credential should be immortal but having overlapping credentials can be a management (synchronization) problem even though overlap is the simplest model for avoiding service interruption at the moment

of expiry. Since the holder of a "before" fragment and an "after" fragment can perform his partial signature with both fragments, were he to pass along both results he would ensure that a credential is valid regardless of whether the first partial signature was with a corresponding before or after fragment. In this way, only the first holder of a chain of partial signatures need know which to use or, more interestingly, no one need know the moment of expiry with any precision – this is "start using new fragment" followed by "stop using old fragment" with as much slop in between as is needed. Since laptops are notoriously time de-synchronized, this has immediate application.

11. **Privacy:** I can issue credentials from my smartcard CA that carry no information other than that I issued them. These credentials can bridge between web-of-trust models (like PGP) and hierarchic models (like X.509) inasmuch as I can cooperate with a number of other individuals with whom I share a web of trust. When I issue a credential, anyone who knows my public key out of band can, at least, verify the credential in the usual sense of certificate verification. In other words, since any PK pair is a candidate for certificate issuance, a PK pair in a conventional hierarchy can issue certificates that are part of that hierarchy. However, if a collection of people will issue certificates for each other based on 2-of-n splitting, a MIX network ([5]) is used so that all a surveiller knows is that some member of the group was vouched for by the group without being able to enumerate the group or identify the signers. In this way pseudonymity can be achieved and, for large enough groups, effective privacy as well. Note that this feature is, to some extent, considered a bug by the standardizers (PKIX and X.509) who want certificate chains to have additional extensions saying whether such and such a key has the right to certify others. An extension to this effect appearing in a certificate may, of course, be ignored but it does likely provide a genuine attempt at risk allocation.

12. **Payment:** I share a 2-of-2 fragment with my preferred financial institution, e.g., VISA, and issue a partially signed IOU to the merchant at point of sale. The merchant, having done the same (shared a 2-of-2 fragment with VISA), also partially signs. The merchant sends the order and payment information to VISA where VISA completes both partial signatures. If and only if both signatures complete satisfactorily does the transaction go through. This is much simpler than SET ([23]); each party affiliates with VISA by giving VISA a key fragment doubly encrypted, first in the private key of an accompanying public key certificate and then in VISA's public key. All VISA has to do is (verify and) store these initial submissions. When a sale is made, the customer and the merchant each have to compute one partial signature and VISA has to compute two, the completion of which constitutes a receipt for the actual function VISA provides, namely to buy the transactional risk from the merchant/consumer pair for a consideration. Compare that to the six certificates, multiple, freshly generated secret keys and over a dozen public key operations that SET requires. SET's degree of statelessness is even maintained. In addition, there can be no mismatch between order and receipt as the matching operation is inherent in VISA's purchase of the transaction.

13. **Corporate procurement:** Same as the above but with a 3-of-3 requirement on the buyer side, i.e., corporate audit/finance partial signature, corporate buyer partial signature and VISA's partial signature. Since the corporation might prefer to have spending limits, it could set key fragments a priori for various spending limits and distribute these as a grant of authority to individuals based on rank or trust. The corporate audit/finance key fragment might even be delegated to some particular corporate buyers so that they had 2-of-the-3 of the 3-of-3 quorum at the outset of the transaction.
14. **Automatic validity checking:** The principle in payments and procurement can be extended. Its crux is that validity checking is replaced by correct collaborative completion according to the intent and therefore style by which the original key was split. Note that if I share my key with many institutes, then I can use a pseudorandom function (applied to the name of the institute) in order to determine the institute share (which determines my share as well in a 2-of-2 split). I can also force more than one entity to participate in this validity checking, where the result of validity is made known to everyone without compromising the distributed separation of power for future validations.
15. **Coupons:** I craft a document and compute some operation on it such as a hash. This hash is fixed as (that is, it becomes) the first fragment of a 2-of-2 quorum. I compute the corresponding second fragment and use that second fragment to partially sign the document. As recipient, you can compute the hash and, ipso facto, you can complete the signature. The resulting document is a validated coupon but your signing it proves that you "read" the document inasmuch as you had to process it to acquire the hash. (Regardless of whether this is 2-of-2, the main idea is that the recipient can compute the remaining fragment but only if he "reads" the document he is signing.) This does not require a certificate and the key fragment used for the upstream partial signing is a throwaway. In other words, we are splitting the same key again and again such that the splits are unique but the verification (public) key is constant. The coupons are inherently use-once inasmuch as part of the text hashed to derive the completing fragment can contain a serial number which need be retained by the redemption center against double spending, and nothing more. (The technical binding of content and keys requires that the splits are random; this implies that a random-oracle hash assumption is needed when a fragment is derived such that many (parallel) random splits are indeed secure (e.g. [13])).
16. **Membership lists:** We split a key much like the above, i.e., multiple 2-of-2 splits on the same key. Each 2-of-2 split is used to pre-sign and distribute a document that, in turn, bears the computable wherewithal to complete the signature. The document is a membership form and the completion of the form and a signature to go with it binds the respondent to membership. The blank form, made unique with a membership number, is distributed to a potential member encrypted in that individual's public key as would be any confidential message to a distant counterparty. When the document is returned to headquarters, the complete signature adds the individual to the membership list. From then on, the member will partially sign (using the key

fragment he accumulated during enrollment) and send the partially signed message to headquarters. Headquarters will complete the signature and send the message on to the other members who can jointly and severally verify the authenticity and appropriateness of the message but need only one key to do this, i.e., the public half corresponding to the self-same private key that has been repeatedly split amongst members. This would, for example, provide an automatable way to diminish spam by forcing a star-patterned routing pattern.

17. **Delegation:** I want to delegate something to my assistant. I split my key into a 2-of-2 quorum, retaining fragment A and sending fragment B to my assistant encrypted in her public key. I send an authorization cert, partially signed in fragment A to my assistant, possibly with a time of applicability in the future. My assistant completes the signature with fragment B turning the otherwise useless proto-certificate into a valid certificate. I can do these fragmentations on a per-use basis or I can make this fragmentation permanent to the relationship with my assistant where she might, say, be authorized to read and reply to my mail but not to sign my name to the final budget document. She could, however, return the budget signed partially in fragment B so that I could, if I was satisfied, complete the signature using fragment A thereby making it possible to have her return data to me on an approval basis but with integrity protection built in inasmuch as the "complete the signature" step would fail should there have been an integrity attack. Since sending the key fragment to my assistant requires a decryption operation on her smartcard, it could be arranged that decrypting a key in that manner had the same "never leaves the card" guarantees that key generation on the card is ordinarily assumed to have. (This is a side point but a useful primitive to insist upon when negotiating firmware upgrades to be installed at time of manufacture.) Delegation to a distributed group of assistants is possible as well. In general, delegation can be combined with other applications of threshold cryptography, e.g., with server controlled off-line deferral to control time, and with distributed proxy encryption [18] to route encrypted data to the assistant with the delegated power.
18. **Virtual corporation:** A group of entities need face-to-face negotiation to consummate a deal of some sort. They meet; their respective smart cards hold their particular private key-halves that permit them to link confidentially (via an IPsec VPN, say) to their home organizations. Another unitary smart card representing the virtual corporation of the moment crafts an n-of-n quorum and distributes the fragments to each participant, first signing it then encrypting it in the public key of each participant so that distribution is secure, non-repudiable and specific. That single card also crafts an unsplit key pair and distributes the public half to each participant. As they work, they encrypt their traffic in that public key forcing all traffic to hub through the holder of the private half. The hub logs (and signs) all traffic using the private key-half of the virtual corporation. Note that all traffic is subjected to the decrypt operation even if the result is unreadable due to super-encryption. Such super-encrypted traffic (confidential messages to headquarters via a VPN, say) is opaque to the other participants but it is

still logged against the day it might become necessary for some participant to show that they had said or done thus and so at such and such a moment in time. At the conclusion of the deal, the controlling smartcard is destroyed (and with it the key it held) sealing the log for good. Since the deal will be signed n-of-n, the deal is non-repudiable by each participant as well as demonstratable to a third party adjudicator. If one participant wishes to do so, that participant can also display their traffic as sealed in the log by demonstrating its extraction from that log in a key that that participant controls. The participants, signing n-of-n, can issue (via the controlling smartcard) certificates that authorize or otherwise enable the deal to be effected on, say, multiple exchanges. When it is time for the corporation to cease existence, the controlling smartcard is destroyed thereby ensuring that the facility of its private key dies and, with it, the virtual corporation. This is a powerful construct.

19. **Elective privacy:** An individual can share a 2-of-2 quorum with any entity who holds private data on that individual (we'll call it a repository). When the individual wishes to grant another third party access to the individual's own private data in the repository, the owner drafts a certificate authorizing the third party access to the individual's data. That certificate can be inherently use-once inasmuch as the data repository front-end must sign it upon presentment by the third party before the back-end would honor it but either the front or back ends could retain it for checking against reuse in a spirit much like the way in which an off-line digital coin is retained by (once presented to) a clearing bank as a check against double spending. The management of digital rights in general is an extension of this thread.
20. **Licensure:** A key necessary to use a resource is split between grantor and grantee. When the grantee wants to use the resource, the grantor's half-signed certificate must be completed by the grantee using, presumably, a key half that was issued to the grantee encrypted in the grantee's public key. Of course, the grantee may have shared his half of the quorum with another, but that would now be traceable at least as far as the grantee. The actual holder of the licensed material, which could well be a third party information warehouse, would accept such certificates only when fully signed, of course. This is preferable to a simple license in that it would require the smartcard of the grantee to be present in order for the resource to be used especially if the grantor were to preset its half of the split key to a fixed computation on the public key of the grantee, i.e., if the 2-of-2 quorum had one share fixed in advance by a computation against the public key corresponding to the grantee's smartcard and returned the other share encrypted in that public key. An extension should smartcards be manufactured with a device key pair burned-in seems obvious.
21. **Trial period / promotion:** An authority can be granted first requiring countersignature, i.e., as a 2-of-2 split, but, after favorable performance, the apprentice might be made a journeyman by providing the other half so that the now ennobled individual had a whole key rather than a mere fragment. This can be extended, of course, such that the promotion could be

from "must seek two approvers" to "must seek one other approver" to "can approve on own authority."

22. **Smartcard application loading:** Split a key 2-of-2 and put one fragment and the public half on the smartcard at the time of manufacture. When the holder of the other fragment wishes to put something onto the card, he would partially sign in his fragment and send it to the smartcard. Upon receipt, the smartcard would finish the signature and confirm it with the public key (providing both an authenticity check on the sender and an integrity check on data). This beats a conventional signature inasmuch as another listener not party to public key embedded in the smartcard could not confirm the signature nor actually figure out much about how it was constructed. Substitute "encrypt" for "sign" as appropriate.
23. **Re-keyed cipher without receiver re-key:** In some situations, where the ciphertext stream of a message is the result of encrypting with a rolling member of a 2-of-n quorum, the recipient would have to *complete the encryption* (so to speak) before he could do the decryption, assuming that he had been passed one member of the quorum and a corresponding public key. This beats both embedding a series of data encrypting keys (DEKs) within the message as well as the independent transmission of such a DEK series inasmuch as the recipient would have only one, oddly arranged, decryption process without re-keying, yet the message stream itself would be encrypted with as many keys as the sender desired. (The recipient may be a small device and the rolling of fragments at the sender can be viewed as re-keying which is done by the server only).
24. **Suicide revocation:** A subscriber crafts a suicide note of the form "My key is compromised, revoke your trust in it now," partially signs it with a 2-of-2 split of that very key's private half, and distributes this suicide note widely enough that it is too dispersed to squelch at the moment of need. When that time of need arrives, send the other fragment signed in the certificate's full private key half to all repositories which might have the ability to process the suicide note, including by broadcast (promiscuous) means. Any recipient holding the suicide note can complete the signature proving validity of the need to revoke. Note that this does not require a fixed point of revocation service though one might be handy. Note also that there is no need for a revocation service signature to make the revocation valid though that, too, may be handy. This suicide note becomes the primary evidence rather than the revocation service's signature which mitigates high-availability requirements on the revocation service and also handles the particularly thorny problem of a subscriber whose public key half is embedded in certificates issued by a multiplicity of disparate, non-communicating CAs. Were the subscriber to have sufficient trust in an associate, that associate could be entrusted with the completion (second) fragment so that even the inability of the subscriber to craft the emergency notification could be handled by these same simple means. Finally note that because the suicide note is invalid until its signature is completed, theft of the note does not create a denial of service risk for the subscriber, i.e, no attacker can send the subscriber's suicide note for him.

3 Business Analysis

While market analysis is often done when cryptographic technology is brought to market, it is rarely published. We next analyze the potential market of threshold systems and possible market penetration scenarios, based on the ideas and applications suggested herein.

As best as we can now characterize it, the quorumed splitting of keys creates a primitive boolean logic that is cryptographically enforceable. Specifically, a simple key split is an "AND" operation on the key fragments inasmuch as all of them are necessary and the negation (unavailability) of any one blocks the effect of all the others. In similar vein, quorumed key splits form an "OR" operation inasmuch as when, for an m-of-n split, m-1 have made their contributions any of the remaining n-m are in alternation. Since the presence of an AND and an OR operation generates a boolean logic, it seems fair to characterize what we have here as a fundamental cryptographic logic as in "(generalized) secret sharing scheme" but applied to capabilities. The above is not the only way to generate the logic and for example an AND-OR tree of capability can be generated. Also, capabilities can be expressed by a combination of threshold cryptographic scheme (implicit sharing) and delegation via explicit certificates enabling new capabilities [21].

This key splitting technology seems optimal for multi-hand protocols where it is essential that no intermediate results are valid (or misusable) in and of themselves. They include problems of delegation, accountability, and the binding of arbitrary collections of counterparties into sharply demarcatable virtual corporations. We would argue that the only boundaries that apply in an electronic world are cryptographic ones, i.e., that geography is replaced by cryptography as how one describes what is inside and what is outside some given boundary (of course, we do not dispute that national boarders, language boarders and other such speedbumps can be and will be introduced in cyberspace, but they are quite artificial against the capabilities of information technology whereas cryptography is not!). The above protocols give a much richer vocabulary to "What does this signature actually mean?" This is a key-centric view of the world, rather than a name-centric one.

3.1 Threshold Cryptography Market Penetration Analysis

Let us now discuss the development and business implications of the thesis we are investigating (we note that this is rarely done in cryptographic technology papers, but is quite proper in a paper like ours). The more we ponder the ideas above the more we are convinced of the versatility of key splitting at the user level. We think that key splitting inherently models activities of joint action much more prosaic than the high end uses that motivated their invention. These activities have to be modeled and presented to the user in an intuitive and easy to grasp logic, as part of the user interface. If there is to be any real use of these ideas, a company that would make money from them must be focused on solving small economic problems on the cost side of the customer's ledger rather than being entranced by the generality of the ideas however much that may

be compelling. What, then, might be such applications? The question is what combination of our list of applications make a nucleus of a product line – one that can yield an early deliverable and can be built upon for additional products. Here are four/five alternatives; not necessarily optimal, not necessarily complete and not necessarily in order by money-making potential. They are:

(0) Split key cryptographic toolkit
(1) Personal CA and countersignature package
(2) Virtual corporation as facilitator for new trading paradigm
(3) Software licensing for metered use
(4) Data recovery form of key escrow

Let us review these application areas:

(0) Split key cryptographic toolkit. A toolkit for split key cryptography in and of itself is (almost) a byproduct of developing ultra-high assurance applications such as root certifying authorities. The target customers might be builders of smartcard operating systems, builders of high assurance systems in general, payment systems integrators and other financial processors, access control systems and government integrators. Thinking like Machiavelli, one would want either to make money off this or, if not, at least make sure that no one else could make any money either (a good quality reference implementation for free, say). If it is to be a free-standing business, the toolkit would have to be a money maker, per se. A team that supplies a strictly for-profit toolkit would have regular conflict between (a) minimizing the separation between the current rev of the product and the state-of-the-art versus (b) maximizing the quality and margin of the current rev. If the principle aim of the endeavor is to get split-key ideas adopted then giving them away gains in attractiveness.

(1) Personal CA and countersignature package. The direct application most likely to be immediately useful, assuming the ability to issue a certificate from a personal (possibly smartcard-based) CA holding a terminal key fragment, is the grouping of countersignature, notarization, authorization and delegation. Being able to control a subordinate's ability to use his superior officer's authority would add value to the role of the subordinate while preventing misuse of the superior officer's authority, as the example of handing a budget for approval up to the titular holder of the signatory key has shown. Delegating the ability to respond to e-mail seems easy to explain to nearly anyone and there are many examples in real life where the formal status of one individual, a licensed architect, say, is used as a cover for many others to do their work only to be signed with the licensed architect's stamp. Delegating power to secretarial staff is another concern in organizations. A first focus, then, would be a personal (assumedly smartcard-resident) function that could unwrap and sequester a key fragment just as surely as several of today's cards can create a key pair within a fundamental guarantee that the freshly created private fragment never leaves the card. Once this is available, the only requirement would be the client functionality (assumedly in the form of browser applets). The rest, i.e., the sign-by-degrees software base, is already in place within a number of high-assurance application code-bases albeit without the finishing touches a mass distribution focus would

require. Should the split-key toolkit materialize as a product, sub-licensable access to it would be the only raw material needed. The business would be to sell a reference implementation to smartcard vendors (with the fall back of at least encouraging a consistency of interface via open source methods), to sell applets to end-users (hard to do), and to sell both to groupware suppliers (subject to strongly organized prior intellectual property protection within the very small number of viable groupware firms). It is probable that participation in standards processes would be essential but, as always, the tradeoff is between making a standard and negotiating one.

(2) Virtual corporation as facilitator for new trading paradigm. Looking to that part of the financial world where technology is a weapon and new ways to package risk are how one takes large quantities of money from the market, bridging deals across exchanges via a virtual corporation is surely the most attractive. Or, more precisely, if you expect to permit single deals to bridge public exchanges, you will need the building blocks of a virtual corporation, along with VPNs and hooks into execution systems, to do so. Extending, in effect, the "key-centric" school of PKI thought, creating a key and then splitting it amongst counterparties is precisely the act of creating a virtual corporation. Where such a corporation is single purpose, unrolling a large and complex position while maintaining hedges in multiple markets say, subsequently destroying the private key half destroys the virtual corporation's ability to craft new signatures yet, by not destroying the public half you preserve the ability of anyone to verify those signatures going forward in time particularly if order execution includes a self-signed instance of that public key. A partner in trading systems support would likely be essential to market success inasmuch as the exchange that can leverage its back-end systems to provide clearance, etc., to bridged, virtual corporation transactions is the exchange that will make other exchanges secondary to it. The business here is to build, on behalf of an exchange, the tools they would need to make this possible, i.e, there does not appear to be a consumer play here. Further thought will be required as to the day trader especially around trust and clearance issues. If those could be finessed, there would be no need to involve existing exchanges though the barriers to entry would be overwhelming should the exchanges decide to embrace the technology while themselves having a piece of the action.

(3) Software licensing for metered use. The model of buying software upgrades at periodic intervals and installing them is under some pressure; something a recent article in the Wall Street Journal called "upgrade fatigue." Of course, some percentage of each upgrade is actually necessary to repair past sins (especially those related to security), but the steady appearance of new features finesses the public discussion that would otherwise ensue as to why they have to pay to have a defect repaired. A universal client (the browser), mobile code in the form of applets and a work force that collectively expects location independent computing press hard on the fixed-cost model that corporate IT investing in per-seat software upgrades represents. If software were rented/metered it could be cheaper in the large, but budget officers prefer fixed price purchases over metered purchases because they favor predictability over lowest possible price.

If software were universally available but only accessible to the holder of the appropriate license, software management would be much cheaper (no tailoring of the desktop needed) while, at the same time, permitting fixed price or metered use to be independently negotiated beyond the issue of who has legitimate access. Licensing to a known individual holding a known smartcard is probably the cleanest answer and has the useful property that the individual could not give away the license without giving away his personal identity. From the individual's point of view, that makes a networked computer in a strange location a useful device inasmuch as the suite of available software would be exactly equivalent to the suite of license fragments on the smartcard inserted into it. The business here would be a license granting engine that is a key-fragment machine, good hooks to on-line transaction systems, some way to reuse (co-opt) existing license services if any such opportunities exist, and the tools to recast this as a species of delegation such that sub-license could also be factored into this mechanism. Barriers to entry would depend on whether the threshold-crypto toolkit were economically reverse-engineer-able (since a closed system need not tell how it does what it does) and the by-now standard pair of relationships and IP protections. Such a model may receive a boost if liability claims against software flaws begin to stick such that the vendor itself would prefer a model of the customer renting the latest, best version rather than continuing to rely on an out-of-date and insecure one.

(4) Data recovery form of key escrow. Finally, the argument that data recovery can be provided without the side effect of enabling surveillance is quite attractive even if for corporations that issue keys (rather than allowing the user to compute them freshly and in private) there is no way to avoid the surveillance threat entirely. Split-key-based data recovery would enable cryptographic filesystems to be more widely deployed and make them operable by the average idiot ("this plastic card is the key and you just put in that keyhole there, just like a hotel room door"). This option has at least the virtue of lazy rollout and no PKI or revocation problem worth talking about. It also means that a standardization of encryption strength and pervasiveness could be done with a single mechanism, i.e., a multi-function excuse for smartcard rollout. The business would be the key fragmentation gear, the smartcard interface and the integration with a cryptographic filesystem, all under intellectual property protections. Making the corporate escrow box pretty resistant would be essential and that might well be the only sine qua non of selling such a simple idea without fear of immediate preemption by the platform vendors and/or OS vendors. The cryptographic filesystems of today are used by the technologically capable more than they are by the slaves of policy, but we can change that. Ordinary people might be interested in this sort of thing for their own purposes ("When I die, Aunt Agnes and my lawyer will be able to open my will"). The general public already has one working example of split key technology in the form of their safety deposit box and the two-key requirement it embodies.

The level of effort to do any of the above alternatives does not seem daunting assuming that a threshold cryptography toolkit [1] is available and that the smartcards/devices become effectively standardized. To the extent that all legacy

applications are coming to have Web front ends, the expression of authorization in the form of role-identity certificates means that the Web infrastructure of a corporation can be used as is. Basic logic which expresses delegation rules, has to be incorporated as a basic intuitive user interface layer. The ability to work with a number of different styles of cryptography is eventually necessary, of course, but the most crucial strategic point is to have something the customer can start with at low commitment cost yet which can be upgraded substantially.

4 Conclusion

Splitting keys is undeniably useful for protecting cryptographic capabilities and increasing their availability. Nevertheless, we now conclude the the main advantage is to decouple from group actions any synchrony constraint yet without producing any intermediate validity (namely atomic distributed operation by a cryptographic service/capability). An incompletely processed split-key signature has no validity and leaks no information about the m or the n. By contrast, full document signatures serially applied (aka multi-signing [3]) create intermediate results that are prima facie valid, which are obviously coordinated and which cannot be made anonymous with respect to what members of the quorum signed. We think that deferring all validity to the atomic moment of signature completion is a very useful building block independent of its anti-fraud application in high assurance systems inasmuch as partial signature leaks no information yet retains integrity protection. We think that to whom an authority is delegated can often be as much a matter of privacy as it is a matter of convenience – a rare juxtaposition. We suspect that the application space of these ideas is much greater than we might now realize. There seems to be a lot here.

References

1. B. Barak, A. Herzberg, D. Naor and E. Shai. The Proactive Security Toolkit and Applications. ACM CCS '99.
2. D. Boneh, X. Ding, G. Tsudik and M. Wong. A Method for Fast Revocation of Public-Key Certificates and Security Capabilities. 10th Usenix '01.
3. C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, eds., *Cryptography and Coding*, Clarendon Press, 1989.
4. R. Cramer, R. Gennaro and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. Eurocrypt '97.
5. D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. CACM Feb '81.
6. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC '94.
7. Y. Desmedt. Society and Group-Oriented Cryptography: A New Concept. Crypto '87.
8. Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89.
9. Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signatures. Crypto '91.

10. Y. Desmedt and S. Jajodia. Redistributing Secret Shares to New Access Structures and Its Applications, manuscript.
11. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen. SPKI Certificate Theory. Internet Network Working Group RFC 2693, Sep. '99.
12. Y. Frankel. A Practical Protocol for Large Group-Oriented Networks. Eurocrypt '89.
13. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. Crypto '97.
14. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptography. FOCS '97.
15. Y. Frankel and M. Yung. "Dynamic Fault"-Robust Cryptosystems Meet Organizational Needs for Dynamic Control. Financial Cryptography '99.
16. S. Garfinkel, PGP: Pretty Good Privacy. O'Reilly, 1994.
17. A. Herzberg, M. Jakobsson, S, Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. CCS '97.
18. M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-encryption, PKC '99.
19. M. Jakobsson and M. Yung. Magic-ink signatures, Eurocrypt '97.
20. L. Lessig. Code: And Other Laws of Cyberspace. Basic Books, 2000.
21. M. Mambo, K Usuda and E. Okamoto. Proxy Signatures for Delegated Signing Operation. ACM CCS '96.
22. R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91.
23. SET Secure Electronic Transaction Specification, Book 3 Protocol Definition, v1.0 '97. http://www.setco.org/download/set_bk3.pdf

Redistribution of Mechanical Secret Shares

Yvo Desmedt[1], Rei Safavi-Naini[2], and Huaxiong Wang[3]

[1] Department of Computer Science, Florida State University, USA
Dept. of Mathematics, Royal Holloway, University of London, UK
desmedt@cs.fsu.edu
[2] School of IT and CS, University of Wollongong, Australia
rei@uow.edu.au
[3] Department of Computing, Macquarie University, Australia
hwang@comp.mq.edu.au

Abstract. Vaults are used extensively in the financial world. Some of these vaults use a secret sharing scheme in which the shares are mechanical keys. Reorganization of a corporation sometimes requires to change the access structure of those authorized to open the vault. Although changing access structure is studied in the context of secret sharing schemes, the techniques are inadequate in the case that the shares are mechanical keys. For example, some schemes require that an existing secret sharing scheme (vault in our case) be fitted with new sets of shares (mechanical keys in our case). That is a number of share sets (key sets) be produced that open the same vault. Making such a modification to a mechanical vault is very expensive, if at all possible. We study how one can redistribute secret shares only by using copying of these shares, which is the only operation one can allow to deal with mechanical shares without changing the mechanical vault mechanism.

Keywords: secret sharing, combinatorics, cover-free family

1 Introduction

The downsizing of dot-com's has made several investors question the importance of e-banking, e-cash, etc. Moreover, recent terrorist attacks have demonstrated the need for protection of our mechanical assets. In the light of these events we question whether the recent research in cryptography has overemphasized the electronic aspect of security.

Although our financial world has become heavily computerized, large amounts of old-fashioned cash are still around and need protection. Moreover several financial institutes (such as banks) offer vaults to clients to store valuables such as jewelry, stocks and bonds. Cyber crime often receives a lot of press, but financial institutes are still vulnerable to old-fashioned theft. An example of a $13.7 million theft is the case of Charlotte [11] in 1998. What is special about this theft is that it was done by a *single* insider who had access to the vault.

To avoid such thefts financial institutes usually require that two keys given to two different people are needed to open a vault. From a cryptographic aspect

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 238–252, 2003.

the security of a single vault is a 2-out-of-2 secret sharing scheme [4,19]. Note that often individual safes are in a vault room behind a locked main armored door where two keys given to two different people are needed to open this main armored door. *Such measures clearly generate more complex access structures.* Although access structures of mechanical secret sharing schemes are equivalent to data based ones, there are major differences with secret sharing schemes in the usual cryptographic setting. When shares are data one may have, for example, homomorphic secret sharing schemes [2]. Such algebraic operations on secret shares have been useful for a broad number of applications, such as threshold cryptography. Therefore, one may conclude that cryptography cannot play a major role on addressing mechanical security, e.g. mechanical secrets' shares. However, in this paper we will demonstrate that techniques used extensively in cryptography can sometimes be used for mechanical security.

We believe that applying cryptography to security problems outside the internet is not only an interesting academic exercise, but recent terrorist attacks have clearly demonstrated the need for protecting against the threats outside the cyber world and the need to examine the usefulness of our techniques in a broader context. In this paper we focus on mechanical secret sharing which is commonly used in financial institutions.

We consider the problem of how a financial institute can reorganize the access structure to a mechanical secret sharing scheme. To be precise, given shares of an ℓ-out-of-v mechanical secret sharing scheme, how can we reorganize it into a t-out-of-n one. One approach is to redesign the safe mechanism, which is quite expensive. Therefore, we study how to achieve this goal without such a redesign. Here, we can only work with the shares that this time are keys without algebraic properties! The only "operation" we allow on mechanical shares is that these are copied. Note that prior work on redistribution of secret sharing, such as [9,1,13, 10,17] is not applicable to mechanical share redistribution.

In this paper we will first discuss in Section 3 our approach. We will see that our problem is equivalent to the construction of what we call a strong cover-free family, a special type of cover-free families that is a concept introduced by Erdös-Frankl-Furedi [12]. We will see that several concepts that have been used extensively in modern cryptography, such as universal hash-functions, can be used to construct strong cover-free families (see Section 4). To evaluate how good our constructions are, in Section 5 we discuss bounds on strong cover-free families. We conclude in Section 6.

2 Preliminaries

This section can be skipped by the reader familiar with secret sharing.

Definition 1. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a group of n participants and let $\mathcal{K}$ denote the set of secrets. We assume P_i's share is selected from the set $\mathcal{S}_i$. A (t, n)-threshold scheme (or also called a t-out-of-n secret sharing scheme) is a pair of algorithms: the* dealer algorithm $\mathcal{D}$ *and the* combiner algorithm $\mathcal{C}$. *For a*

secret from $\mathcal{K}$ and a randomly chosen element of $\mathcal{R}$, the dealer algorithm applies the mapping

$$\mathcal{D} : \mathcal{K} \times \mathcal{R} \to \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$$

to assign shares to participants in $\mathcal{P}$. The combiner algorithm takes the shares of a subset $B \subseteq \mathcal{P}$ of participants and if the set $B \subseteq \mathcal{P}$ and $|B| \geq t$ it returns the secret.

$$\mathcal{C} : \bigcup_{P_i \in A} \{\mathcal{S}_i\} \to \mathcal{K}.$$

$\mathcal{K}, \mathcal{S}_i$ corresponds random variables $\mathbf{K}$ and $\mathbf{S}_i$. A secret sharing scheme is perfect if for all $(i_1, i_2, \ldots, i_{t-1})$ where $1 \leq i_j \leq n$:

$$\text{prob}(\mathbf{K} = k \mid \mathbf{S}_{i_1} = s_{i_1}, \mathbf{S}_{i_2} = s_{i_2}, \ldots \mathbf{S}_{i_{t-1}} = s_{i_{t-1}}) = \text{prob}(\mathbf{K} = k).$$

The most known (t, n) secret sharing scheme using algebra is Shamir's scheme [19]. Since we focus on mechanical secret sharing, we do not survey it.

One measure for efficiency of the secret sharing scheme can be through the notion of share expansion.

Definition 2. *Under the above notation, we define the share expansion of a secret share scheme as*

$$\rho = \max_{1 \leq i \leq n} \frac{\log |\mathcal{S}_i|}{|\mathcal{K}|}.$$

Throughout this paper, all logarithms are to the base 2, unless otherwise indicated.

3 Our Approach

We first give an informal description of how to reorganize a mechanical secret sharing scheme. We then discuss this formally.

We assume we have a mechanical ℓ-out-of-v perfect secret sharing scheme $(\mathcal{D}_0, \mathcal{C}_0)$ and we want to reorganize it into a perfect t-out-of-n $(\mathcal{D}, \mathcal{C})$ one. Assume that shareholder P_i' $(1 \leq i \leq v)$ in $(\mathcal{D}_0, \mathcal{C}_0)$ has a mechanical share a_i in the ℓ-out-of-v secret sharing scheme. We now need to give a share s_j to shareholder P_j $(1 \leq j \leq n)$ in $(\mathcal{D}, \mathcal{C})$. Since the only allowable operation is to make copies of a_i, s_j can only be a collection of mechanical shares. So s_j can be described using a set B_j of indexes of shares a_i of which P_j received a copy. To guarantee that the resulting $(\mathcal{D}, \mathcal{C})$ sharing scheme is a t-out-of-n one, we need

for the combiner algorithm that if t shareholders P_j put their shares together, they jointly have ℓ mechanical shares of the old $(\mathcal{D}_0, \mathcal{C}_0)$ scheme.

to guarantee perfectness that if $t-1$ shareholders P_j put their shares together, they jointly have at the most $\ell - 1$ shares of the old $(\mathcal{D}_0, \mathcal{C}_0)$ scheme.

We now describe this formally. We first introduce the new concept of a strong cover-free family, a special case of a cover-free family [12] (as proven in Theorem 12).

Definition 3. *A* strong cover-free family *is a set system* $(X, \mathcal{B})$ *such that the following properties are satisfied:*

1. $X = \{x_1, \ldots, x_v\}$ *called* points*;*
2. $\mathcal{B} = \{B_1, \ldots, B_n\}$ *is a set of* n *subsets of* X*, called* blocks *(*$B_i \subseteq X$*);*
3. *For any* Δ *and any* $\Lambda \subseteq \{1, \ldots, n\}$ *with* $|\Delta| = t$ *and* $|\Lambda| = t - 1$*:*

$$|\cup_{i \in \Delta} B_i| > |\cup_{j \in \Lambda} B_j|. \tag{1}$$

We will call $(X, \mathcal{B})$ *a* (v, n, t)*-strong cover-free family (or* $(v, n, t) - SCFF$ *for short).*

The idea behind our new construction is to combine an existing threshold scheme and a SCFF to construct a new threshold scheme. The construction works as follows.

1. Assume $(X, \mathcal{B})$ is a (v, n, t)-SCFF. Let ℓ be a integer such that $\min_{\Delta} | \cup_{i \in \Delta} B_i| \geq \ell > \max_{\Lambda} | \cup_{i \in \Lambda} B_i|$ where Δ runs through all t-subsets of $\{1, \ldots, n\}$ and Λ runs through all $(t-1)$- subsets of $\{1, \ldots, n\}$. Since $(X, \mathcal{B})$ is a SCFF, such ℓ exists.
2. Assume there is a (ℓ, v) threshold scheme $(\mathcal{D}_0, \mathcal{C}_0)$. For a secret $k \in \mathcal{K}$, the v shares of $(\mathcal{D}_0, \mathcal{C}_0)$ are $a_1, \ldots, a_v$.
3. Define a (t, n) threshold scheme for n participants $P_1, \ldots, P_n$ by constructing n shares $\mathbf{s}_1, \ldots, \mathbf{s}_n$ as the ordered (ordered by their index) multiset $\mathbf{s}_i = \{a_j \mid \text{if and only if } x_j \in B_i\}$ and assigning $\mathbf{s}_i$ to the participants P_i for all $1 \leq i \leq n$.

We denote the resulting (t, n) scheme as $(\mathcal{D}, \mathcal{C})$ and prove the following result.

Theorem 1. *If* $(\mathcal{D}_0, \mathcal{C}_0)$ *is perfect, then* $(\mathcal{D}, \mathcal{C})$ *is perfect.*

Proof. Clearly, any t participants, $P_{i_1}, \ldots, P_{i_t}$ say, have $|\mathbf{s}_{i_1} \cup \cdots \cup \mathbf{s}_{i_t}| \geq \ell$ shares from the v share of the (ℓ, v) threshold scheme $(\mathcal{D}_0, \mathcal{C}_0)$. From the choice of ℓ, we know that t participants can reconstruct the secret by applying the combiner algorithm $\mathcal{C}_0$. Next, any $t-1$ participants have no extra information about k provided $(\mathcal{D}_0, \mathcal{C}_0)$ is perfect. Indeed, assume that $P_{i_1}, \ldots, P_{i_{t-1}}$ want to recover the secret by using their shares $\mathbf{s}_{i_1} \cup \cdots \cup \mathbf{s}_{i_{t-1}} \subseteq \{a_1, \ldots, a_v\}$. Since $|\mathbf{s}_{i_1} \cup \cdots \cup \mathbf{s}_{i_{t-1}}| < \ell$ and the underlying (t, v) scheme $(\mathcal{D}_0, \mathcal{C}_0)$ is perfect, the claim follows.

Note that the share expansion ρ of $(\mathcal{D}, \mathcal{C})$ is determined by the share expansion ρ_0 of $(\mathcal{D}_0, \mathcal{C}_0)$ and the parameters of the (v, n, t)-SCFF $(X, \mathcal{B})$. We have $\rho \leq \max_{1 \leq i \leq n} |B_i| \rho_0$. In particular, if $(\mathcal{D}_0, \mathcal{C}_0)$ is *ideal*, i.e. $\rho_0 = 1$ and $|B_i| = r$ for all $1 \leq i \leq n$, then $\rho = r$.

3.1 An Example

It is well known that it is easy to make a $(\ell-1)$-out-of-$(v-1)$ perfect secret sharing scheme from an ℓ-out-of-v one. Indeed, give P_i $(1 \leq i \leq v-1)$ as share $s_i = \{a_i, a_v\}$, where a_i is P_i' share in the ℓ-out-of-v secret sharing scheme. So, this corresponds to a $(v, v-1, \ell-1)$-strong cover-free family.

It is obvious that given a mechanical ℓ-out-of-v perfect secret sharing scheme one cannot redistribute the shares to obtain a t-out-of-n one where $t > \ell$. Indeed, let $\Delta \subseteq \{1, \ldots, n\}$ such that $|\Delta| = t$ and $\Lambda \subseteq \Delta$ such that $|\Lambda| = t-1$. To guarantee perfectness the $t-1$ shareholders in Λ should have at maximum $\ell-1$ shares of the old scheme, so at least $t-\ell$ shareholders in Λ need empty sets B. Let us call Λ' those shareholders in Δ that received an empty set. Then, the shareholders in $\Delta \setminus \Lambda'$ can construct the secret key. However, since $|\Lambda'| \geq t-\ell$ we now have that $|\Delta \setminus \Lambda'| = t - |\Lambda'| \leq \ell < t$ can reconstruct the secret. So, we have a contradiction.

We discuss other such bounds in Section 5. We now discuss constructions.

4 Constructions

The following lemma is essential for most of our later constructions.

Lemma 1. *Let $(X, \mathcal{B})$ be a set system such that*

1. $|B_i| = r$, *for all* $i \in \{1, \ldots n\}$;
2. $|B_i \cap B_j| \leq \mu$, *for all* $i \neq j \in \{1, \ldots, n\}$.

Then $(X, \mathcal{B})$ is a (v, n, t) SCFF provided $\binom{t}{2} < r/\mu$.

Proof. Let Δ and Λ be two subset of $\{1, \ldots, n\}$ such that $|\Delta| = t$ and $|\Lambda| = t-1$. We have $|\cup_{i\in\Delta} B_i| \geq \sum_{i\in\Delta} |B_i| - \sum_{i,j\in\Delta} |B_i \cap B_j| \geq tr - \binom{t}{2}\mu = (t-1)r + (r - \binom{t}{2}\mu) >$ $(t-1)r = \sum_{j\in\Lambda} |B_j| \geq |\cup_{j\in\Lambda} B_j|$.

For a given $(v, n, t) - SCFF$ we know that the new mechanical secret sharing scheme is a t-out-of-n threshold scheme. We call the old scheme a ℓ-out-of-v one. For many of the schemes based on the Lemma 1 one could choose ℓ between

- $(t-1)r + 1$ and
- $tr - \binom{t}{2}\mu$,

as is obvious from the proof of Lemma 1. Since for many of our schemes, it is trivial to compute these values, we leave it to the reader.

Note however, we do not have an efficient algorithm to compute the largest possible ℓ with an $(v, n, t) - SCFF$ (i.e. $\min_{\Delta, |\Delta|=t} |\cup_{i\in\Delta} B_i| - 1$) or the smallest one (i.e. $\max_{\Lambda, |\Lambda|=t-1} |\cup_{i\in\Lambda} B_i|$).

4.1 Constructions from Combinatorial Designs

In this subsection, we will give some constructions of SCFF from certain combinatorial designs, including μ-designs, packing designs and orthogonal array. Similar constructions for traceability schemes and frameproof codes can be found in [21].

As before, we will use $(X, \mathcal{B})$ to denote a set system in which X is a finite set and $\mathcal{B}$ is a family of subsets of X. The elements of X and $\mathcal{B}$ are called *points* and *blocks*, respectively. A $\mu - (v, r, \lambda)$ *design* is a set system $(X, \mathcal{B})$, where $|X| = v, |B| = r$ for every $B \in \mathcal{B}$, and every μ-subset of X occurs in *exactly* λ blocks in $\mathcal{B}$. We will only be interested in $\mu - (v, r, 1)$ design.

Theorem 2. *If there exists a* $\mu - (v, r, 1)$ *design, then there exists a* $(v, \binom{v}{\mu}/\binom{r}{\mu}, t)$*-SCFF for any* t *satisfying* $\binom{t}{2} < \frac{r}{\mu - 1}$.

Proof. It is well known that in a $\mu - (v, r, 1)$ design the number of blocks n is exactly $\binom{v}{\mu}/\binom{r}{\mu}$. Assume there exists a $\mu - (v, r, 1)$ design $(X, \mathcal{B})$. Then for each pair $B_i, B_j \in \mathcal{B}$, we trivially have $|B_i \cap B_j| \leq \mu - 1$. From Lemma 1 the theorem is immediate.

There are many results on existence and constructions of $\mu - (v, r, 1)$ design for $r = 2, 3$ [6]. On the other hand, no $\mu - (v, r, 1)$ design with $v > r > \mu$ is known to exist for $\mu \geq 6$. Furthermore, it is known that for $3 \leq r \leq 5$, a $2 - (v, r, 1)$ design exists if and only if $v \equiv 1$, or $r \bmod (r^2 - r)$. To apply Theorem 2, it is required $r \geq 4$ and so that $\binom{3}{2} < r/(\mu - 1)$, where $\mu = 2$. Since $2 - (v, 4, 1)$ design exists for any $v \equiv 1, 4 \bmod 12$, Theorem 2 yields the following result.

Corollary 1. *There exists* $(v, \frac{v(v-1)}{12}, 3) - SCFF$ *for all* $v \equiv 1, 4 \bmod 12$.

A $\mu - (v, r, \lambda)$ *packing design* is a set system $(X, \mathcal{B})$, where $|X| = v, |B| = r$ for every $B \in \mathcal{B}$, and every μ-subset of X occurs in *at most* λ blocks in $\mathcal{B}$. Similar to Theorem 2, we have the following theorem.

Theorem 3. *If there exists a* $\mu - (v, r, 1)$ *packing design having* n *blocks, then there exists a* $(v, n, t) - SCFF$ *if* $\binom{t}{2} < \frac{r}{\mu - 1}$.

As we noted previously, no $\mu - (v, r, 1)$ designs are known to exist if $v > r > \mu \geq 6$. However, for any μ, there are infinite classes of packing designs with a "large" number of blocks (i.e. close to $\binom{v}{\mu}/\binom{r}{\mu}$). Such packing designs can also be constructed from orthogonal arrays. Recall [6] that an *orthogonal array* $OA(\mu, r, s)$ is a $r \times s^{\mu}$ array, with entries from a set of $s \geq 2$ symbols, such that in any μ rows, every μ-tuple with elements from s occurs in these μ rows as an $\mu \times 1$ column vector exactly once. We now use this to prove the following theorem.

Corollary 2. *For any prime power* q *and any integer* $\mu < q$, *there exist* $(q^2 + q, q^{\mu}, t) - SCFF$ *for any* t *satisfying* $\binom{t}{2} < \frac{q+1}{\mu - 1}$.

Proof. In [21], Stinson and Wei showed that if there is an $OA(\mu, r, s)$, then there is a $\mu - (rs, r, 1)$ packing design that contains s^μ blocks. It is well known that for any prime power q with $\mu < q$, there exists an $OA(\mu, q+1, q)$ [6]. It follows that there exists a $\mu - (q^2+q, q+1, 1)$ packing design $(X, \mathcal{B})$. From Theorem 3, we have the theorem.

4.2 Constructions from Universal Hashing Families

The concept of *universal hashing family* was invented by Carter and Wegman [8] and has been the foundations of several applications (see e.g. [24,18]).

Let $\epsilon > 0$. A multiset H of N functions from a n-set X to a m-set Y is called *ϵ-almost universal* ($\epsilon - AU$ for short) if for any two distinct elements $x_1, x_2 \in X$, there exists at most ϵN functions $h \in H$ such that $h(x_1) = h(x_2)$. Without loss of generality we will assume that $n \geq m$. We call H an $\epsilon - AU(N; n, m)$ hashing family. The following shows that SCFF can be constructed from AU hashing families.

Theorem 4. *If there exists an $\epsilon - AU(N; n, m)$ hashing family, then there exists a (Nm, n, t)-SCFF provided $\binom{t}{2} < 1/\epsilon$.*

Proof. Assume that H is an $\epsilon - AU(N; n, m)$ hashing family from S to T. We construct a set system $(X, \mathcal{B})$ as follows. Set $X = H \times T = \{(h, t) \mid h \in H, t \in T\}$ and $\mathcal{B} = \{B_s \mid s \in S\}$, where for each $s \in S$ we define $B_s = \{(h, h(s)) \mid h \in H\}$. Then it is easy to see that $|X| = Nm$, $|\mathcal{B}| = n$ and $|B_s| = N$ for each $s \in S$. For each pair $B_s, B_{s'} \in \mathcal{B}$, we have

$$\begin{aligned} |B_s \cap B_{s'}| &= |\{(h, h(s)) \mid h \in H\} \cap \{(h, h(s')) \mid h \in H\}| \\ &= |\{h \mid h(s) = h(s'), h \in H\}| \\ &\leq \epsilon N \end{aligned}$$

From Lemma 1, we know that $(X, \mathcal{B})$ is an (Nm, n, t) SCFF if $\binom{t}{2} < \frac{N}{\epsilon N} = \frac{1}{\epsilon}$, and the desired result follows.

$\epsilon - AU$ are strictly related to error-correcting codes (see [3]). Let Y be an alphabet of q symbols. An (N, M, D, q) *code* is a set $\mathcal{C}$ of M vectors in Y^N such that the Hamming distance between any two distinct vectors in $\mathcal{C}$ is at least D. The code is *linear* if q is a prime power, $Y = GF(q)$, and $\mathcal{C}$ is a subspace of the vectorspace $GF(q)^N$. Then we will denote it by an $[N, m, D, q]$ code, where $m = \log_q M$ is the *dimension* of the code.

Let $\mathcal{C}$ be a (N, M, D, q) code, we can define a family of functions $H = \{h_1, \ldots, h_N\}$ from $\mathcal{C}$ to Y by the following

$$(h_1(v), h_2(v), \ldots, h_N(v)) = (v_1, \ldots, v_N)$$

for each $v = (v_1, \ldots, v_N) \in \mathcal{C}$.

The following equivalence is due to Bierbrauer, Johansson, Kabatianskii and Smeets [3].

Theorem 5 ([3]). *If there exists an (N, M, D, q) code, then there exists a $(1 - \frac{D}{N}) - AU(N; M, q)$ hash family. Conversely, if there exists an $\epsilon - AU(N; n, m)$ hash family, then there exists an $(N, n, N(1-\epsilon), m)$ code.*

If we apply the above theorem to Justesen codes (Theorem 9.2.4 [23]), we obtain a (v, n, t)-SCFF $(X, \mathcal{B})$ with $|B_i| = O(\log n)$, for all $B_i \in \mathcal{B}$, and the share expansion of the new scheme in our construction is $O(\log n)$.

Another application of Theorem 5 is to use Reed-Solomon codes. An *extended Reed-Solomon* code is a linear code having parameters $[q, u, q-u+1, q]$, where $u \leq q$ and q is a prime power. Applying Theorem 4 and 5 we have

Corollary 3. *Let q be a prime power and $1 \leq u \leq q$. There exists a (q^2, q^u, t) SCFF, where $t \leq \sqrt{\frac{2q}{u-1}} + 1$.*

Proof. Applying the extended Reed-Solomon codes in Theorem 5, we know that there is a $\frac{u-1}{q} - AU(q, q^u, q)$ hashing family. The result follows immediately from Theorem 4.

Using a recursive construction, Stinson (Theorem 6.1 [20]) proved that there exists an $i/q - AU(q^i; q^{2^i}, q)$ hashing family. This in conjunction with with Theorem 4 gives us an infinite class of SCFFs.

Corollary 4. *Let q be a prime power and let $i \geq 1$ be an integer. Let $t \leq \sqrt{\frac{2q}{i}} + 1$. Then there exists a (q^{i+1}, q^{2^i}, t)-SCFF.*

4.3 Construction Based on Exponential Sums

In [14] Helleseth and Johansson used exponential sums over finite fields to construct strongly universal hashing families and authentication codes, motivated by the universal construction in the previous subsection we show that the exponential sums can be applied to our constructions of SCFF with good parameters.

Let $GF(q)$ be a finite field with characteristic p, and let $Tr_{q^m/q}(\alpha)$ be the trace function from $GF(q^m)$ to $GF(q)$ defined by

$$Tr_{q^m/q}(\alpha) = \alpha + \alpha^q + \cdots + \alpha^{q^{m-1}}.$$

Lemma 2 ([14]). *Let $f(x) = \sum_{i=1}^{D} a_i x^i \in GF(q^m)[x]$ be a polynomial of degree D that is not expressed in the form $f(x) = g(x)^p - g(x) + \theta$ for any $g(x) \in GF(q^m)[x], \theta \in GF(q^m)$. Let*

$$N_\alpha(f) = |\{x \in GF(q^m) \mid Tr_{q^m/q}(f(x)) = \alpha\}|.$$

Then $N_\alpha(f) \leq q^{m-1} + (D-1)\sqrt{q^m}$ for any $\alpha \in GF(q)$.

Let $D \leq \sqrt{q^m}$, we define a set $\mathcal{F}_D$ of polynomials with degree less than or equal to D by

$$\mathcal{F}_D = \{f(x) \mid f(x) = a_1 x + a_2 x^2 + \cdots + a_D x^D \in GF(q^m)[x], a_i = 0, \text{ whenever } p \mid i\}.$$

Then, $\mathcal{F}_D$ is clearly a $(D - \lfloor D/p \rfloor)$-dimensional vector space over $GF(q^m)$, so we have $|\mathcal{F}_D| = q^{m(D-\lfloor D/p \rfloor)}$. Moreover, it is easy to see that for each $f(x) \in \mathcal{F}_D$, $f(x)$ can not be expressible in the form $f(x) = g(x)^p - g(x) + \theta$, and Lemma 2 can be applied. That is, for each $f(x) \in \mathcal{F}_D$ and $\alpha \in GF(q)$, we have $N_\alpha(f) \leq q^{m-1} + (D-1)\sqrt{q^m}$.

For each $\beta \in GF(q^m)$, we associate a function g_β from $\mathcal{F}_D$ to $GF(q)$ defined by $g_\beta(f) = Tr_{q^m/q}(f(\beta))$.

Denote $\mathcal{G} = \{g_\beta \mid \beta \in GF(q^m)\}$. Let $X = \mathcal{G} \times GF(q)$ and $\mathcal{B} = \{B_f, f \in \mathcal{F}_D\}$, where $B_f = \{(g_\beta, g_\beta(f)) \mid \beta \in GF(q^m)\}$. We will show that such set systems $(X, \mathcal{B})$ are SCFF with some appropriate parameters.

Lemma 3. *$|B_f \cap B_{f'}| \leq q^{m-1} + (D-1)\sqrt{q^m}$, for any $f \neq f' \in \mathcal{F}_D$.*

Proof. As is easy to verify:

$$\begin{aligned} |B_f \cap B_{f'}| &= |\{g_\beta \mid g_\beta(f) = g_\beta(f')\}| \\ &= |\{\beta \mid Tr_{q^m/q}(f(\beta)) = Tr_{q^m/q}(f'(\beta))\}| \\ &= |\{\beta \mid Tr_{q^m/q}(f - f')(\beta) = 0\}| \\ &\leq N_0(f - f') \\ &\leq q^{m-1} + (D-1)\sqrt{q^m} \end{aligned}$$

Combining Lemma 3 and Lemma 1, we have the following result.

Theorem 6. *$(X, \mathcal{B})$ is a $(q^{m+1}, q^{m(D-\lfloor D/p \rfloor)}, t) - SCFF$ provided $\binom{t}{2} \leq q^m/(q^{m-1} + (D-1)\sqrt{q^m})$*

The above theorem results in an infinite class of SCFFs with good parameters. For example, taking $D = q^{m/2-1} + 1$ for any even m, then for any t satisfying $\binom{t}{2} \leq q/2$, applying the above theorem gives a $(q^m, q^{m(q^{m/2-1} - \lfloor q^{m/2-1}/p \rfloor)}, \lfloor \sqrt{2} q^{m/4} \rfloor) - SCFF$ for all even m. A simple approximation yields that there is an infinite class of $(v, n, t) - SCFFs$ in which the parameters satisfy $\log n = C\sqrt{v} \log v$, where C is a fixed constant. We have $(\log n)^2 = C^2 v (\log v)^2 \geq c^2 v$, that is, there exists an infinite class of $(v, n, t) - SCFF$ in which v is $O((\log n)^2)$.

4.4 Constructions Based on Arcs

An arc is a set system $(X, \mathcal{B})$ such that the intersection of any two blocks (lines) is a single point and the intersection of three blocks is empty [15]. This gives the following result not based on Lemma 1.

Theorem 7. *For each prime power q and each t where $2 \leq t \leq q+1$, there exists an $(q^2 + q + 1, q + 1, t)$-SCFF.*

Proof. As is well known [15] a projective plane allows an $(q^2+q+1, q+1)$ block system as an arc. The blocks are the lines in the arc. So from the definition of an arc we immediately have that the union of a (where $a \leq q+1$) different blocks is exactly $a(q+1) - \binom{a}{2}$ since any two lines have an intersection, but three (or more) blocks of the arc never have. So, t blocks have strictly more points than $t-1$ blocks have, since $t(q+1) - \binom{t}{2} > (t-1)(q+1) - \binom{t-1}{2}$ because $(q+1) > (t(t-1) - (t-1)(t-2))/2 = (t-1)$ which follows from our conditions on t.

This theorem can be extended using n-arcs [25]. An n-arc is a set of n points in the projective geometry $PG(k-1, q)$ such that any k points are linearly independent. An SCFF can be constructed from an n-arc in a way similar to above. parameters of this SCFF will be given in the final version of the paper.

We note that the problem of finding a $[n, k, n-k+1]$ maximum distance separable code is equivalent to the problem of finding an n-arc in the projective geometry $PG(k-1, q)$ and it is known that $n \leq q+k-2$.

4.5 Constructions Based on the Complement

We now discuss SCFF based on applying De Morgan's law to the definition. For a set $A \subseteq X$ we define the complement as $\bar{A} = X \setminus A$.

Lemma 4. *A set system $(X, \mathcal{B}')$, where $\mathcal{B}' = \{B'_1, \ldots, B'_n\}$, for which for all $\Delta \subseteq \{1, \ldots, n\}$ and all $\Lambda \subseteq \{1, \ldots, n\}$ where $|\Delta| = t$ and $|\Lambda| = t-1$ we have*

$$|\cap_{i \in \Delta} B'_i| < |\cap_{i \in \Lambda} B'_i| \tag{2}$$

induces an $(|X|, n, t)$-SCFF $(X, \mathcal{B})$ by taking $B_i = \bar{B}'_i$.

Proof. Suppose that the conditions in the lemma are satisfied. Now take the complement of the left and right hand side of (2) and apply De Morgan's law. It is then easy to see that we obtain (1) in which $B_i = \bar{B}'_i$. The lemma then follows immediately.

We are now discussing SCFF with $\ell = v$. This will imply that we can construct a t-out-of-n secret sharing scheme from a v-out-of-v scheme.

Theorem 8. *Constructing a (t, n)-threshold scheme from a (v, v) scheme using an (v, n, t)-SCFF is equivalent to the following conditions on B_i:*
For any Δ and any $\Lambda \subseteq \{1, \ldots, n\}$ with $|\Delta| = t$ and $|\Lambda| = t-1$:

$$\bigcap_{i \in \Delta} \bar{B}_i = \emptyset \qquad \text{and} \qquad \bigcap_{i \in \Lambda} \bar{B}_i \neq \emptyset$$

where $\bar{B} = X \setminus B$.

Proof. It is easy to see that the addressed problem of constructing a (t, n) scheme from a (v, v) scheme is equivalent to requiring that for any Δ and any $\Lambda \subseteq \{1, \ldots, n\}$ with $|\Delta| = t$ and $|\Lambda| = t - 1$:

$$\bigcup_{i \in \Delta} B_i = X \quad \text{and} \quad \bigcup_{i \in \Lambda} B_i \neq X. \tag{3}$$

The rest follows from Lemma 4.

It is now obvious that an arc can be used to design such SCFFs. This gives the following corollary.

Corollary 5. *For each prime power q, there exists an $(q^2+q+1, q+1, 3)$-SCFF that allows to construct a $(3, q+1)$-threshold scheme from a (q^2+q+1, q^2+q+1) scheme.*

Proof. From the definition of an arc we have that any 2 lines of the arc always intersect in one point, but the intersection of any 3 lines of the arc is empty. The rest follows from Theorem 8 and the well known [15] property that a projective plane allows an $(q^2 + q + 1, q + 1)$ block system as an arc.

We now use Lemma 4 to obtain other SCFFs.

Theorem 9. *Any $\mu - (n, r, 1)$-design gives us a $(\binom{n}{\mu}/\binom{r}{\mu}, n, \mu)$-SCFF.*

Proof. Let us take an $\mu-(n, r, 1)$ design $(X', \mathcal{B}')$ (see Section 4.1 for a definition). As is well known, this implies we have exactly $\binom{n}{\mu}/\binom{r}{\mu}$ blocks. A block system where $|X'| = n$ and $|\mathcal{B}'| = a$ corresponds with an $a \times n$ incidence matrix M over $GF(2)$. So, the transpose of M induces a block system where $(X, \mathcal{B}) = (\mathcal{B}', X')$. Now let $a = \binom{n}{\mu}/\binom{r}{\mu}$.

The definition of the design implies that every μ-subset of X' occurs in *exactly* one block in $\mathcal{B}'$. So, an $\mu - 1$ subset must occur in more than one block in $\mathcal{B}'$. Using the transpose idea and realizing that the columns of M now correspond to blocks in the new $(X, \mathcal{B}) = (\mathcal{B}', X')$ block system this implies that in the new $(X, \mathcal{B}) = (\mathcal{B}', X')$ block system we have that the intersection of μ blocks in the new $(X, \mathcal{B})$ is 1, and the intersection of $\mu - 1$ blocks in the new $(X, \mathcal{B})$ is strictly larger than 1. Using Lemma 4 the theorem follows.

Corollary 6. *This easily extends to $\mu - (n, r, \lambda)$-designs.*

5 Bounds

As noted earlier the efficiency of our new construction relies on the parameters strong cover free family involved in the construction. In this section we will derive some bounds on various parameters of SCFFs.

The following theorem completely characterizes the SCFF when $t = 2$.

Theorem 10. *There exists a* $(v, n, 2) - SCFF$ *if and only if* $n \leq \binom{v}{\lfloor \frac{v}{2} \rfloor}$.

Proof. Assume that $(X, \mathcal{B})$ is a $(v, n, 2) - SCFF$, then it is easy to see that there do not exist two distinct blocks B_i, B_j such that $B_i \subseteq B_j$, i.e., $(X, \mathcal{B})$ is a *Sperner Family* [6]. It is well-known that there exists a Sperner family consisting of n subsets of a v-set only if $n \leq \binom{v}{\lfloor \frac{v}{2} \rfloor}$ (see [6]). Conversely, we can take all $\lfloor \frac{v}{2} \rfloor$-subsets of a v-set, it is easy to see that it results in a $(v, n, 2)$-SCFF with $n = \binom{v}{\lfloor \frac{v}{2} \rfloor}$, proving the desired result.

Next, we derive a lower bound on v for given n and t.

Theorem 11. *In any* $(v, n, t) - SCFF$, *we have* $v \geq (t-1) \log \frac{n}{t-1}$

Proof. Assume that $(X, \mathcal{B})$ is a $(v, n, t) - SCFF$. Let $\mathcal{F} = \{\cup_{i \in \Lambda} B_i : \Lambda \subseteq \{1, \ldots, n\} \mid \text{with } |\Lambda| = t-1, B_i \in \mathcal{B}\}$. We observe that for any $\Lambda \neq \Lambda' \subset \{1, \ldots, n\}$ with $|\Lambda| = |\Lambda'| = t-1$, we have $\cup_{i \in \Lambda} B_i \neq \cup_{j \in \Lambda'} B_j$. Indeed, otherwise assume that there are Λ and Λ' with $|\Lambda| = |\Lambda'| = t-1$ such that $\cup_{i \in \Lambda} B_i = \cup_{j \in \Lambda'} B_j$. Since $\Lambda \neq \Lambda'$, we may assume that there is an element $\ell \in \Lambda'$, but $\ell \notin \Lambda$. It follows $\cup_{i \in \Lambda \cup \{\ell\}} B_i = \cup_{i \in \Lambda'} B_i$ and hence $|\cup_{i \in \Lambda \cup \{\ell\}} B_i| = |\cup_{i \in \Lambda'} B_i|$, which contradicts with the assumption that $(X, \mathcal{B})$ is a $(v, n, t) - SCFF$ since $|\Lambda \cup \{\ell\}| = t$ and $|\Lambda'| = t-1$. So we have $|\mathcal{F}| = \binom{n}{t-1}$. Similarly, it is easy to see that $\mathcal{F}$ is a *Sperner family* [6], that is, for any $F \neq F' \in \mathcal{F}$, we always have $F \not\supseteq F'$. It follows (see [6]) that $|\mathcal{F}| \leq \binom{v}{\lfloor \frac{v}{2} \rfloor}$. Since $\binom{n}{t-1} \geq (\frac{n}{t-1})^{t-1}$ and $\binom{v}{\lfloor \frac{v}{2} \rfloor} \leq 2^v$, we obtain $(\frac{n}{t-1})^{t-1} \leq 2^v$, and so $v \geq (t-1) \log \frac{n}{t-1}$.

The above theorem can be restated as $n \leq (t-1) 2^{\frac{v}{t-1}}$, which gives an upper bound on n for given v and t. We now see that a strong cover-free family is a special case of a cover-free family [12].

Theorem 12. *A* (v, n, t)*-SCFF is a* (v, n, t)*-cover free family.*

Proof. Assume that $(X, \mathcal{B})$ is a (v, n, t)-SCFF. Let Λ be a subset of $\{1, \ldots, n\}$ such that $|\Lambda| = t-1$, and let $i \notin \Lambda$. Then we have $|\Lambda \cup \{i\}| = t$, and so $|\cup_{j \in \Lambda \cup \{i\}} B_j| > |\cup_{j \in \Lambda} B_j|$. It follows that $B_i \not\subseteq \cup_{j \in \Lambda} B_j$, that is, the union of any $t-1$ blocks in $\mathcal{B}$ can not cover any remaining one in $\mathcal{B}$. Such a set system is called (v, n, t)-cover free family [12].

Corollary 7. *Let* $(X, \mathcal{B})$ *be a* $(v, n, t) - SCFF$ *such that* $|B_i| = r$ *for all* $B_i \in \mathcal{B}$. *Then* $n \leq \binom{v}{m} / \binom{r-1}{m-1}$, *where* $m = \lceil r/t - 1 \rceil$.

Proof. Using Theorem 12 and by Proposition 2.1 of [12], the result follows immediately.

We can show that in any SCFF $(X, \mathcal{B})$ where $|X| < |\mathcal{B}|$, the parameter t can not be too large relative to n.

Corollary 8. *In a* $(v, n, t) - SCFF$, *where* $v < n$, *we have* $t < \sqrt{2n}$.

Proof. Indeed, assume that $(X, \mathcal{B})$ is a $(v, n, t) - SCFF$. From Theorem 12 we know that $(X, \mathcal{B})$ is a $(v, n, t-1)$-cover-free family. By Proposition 3.4 of [12], we have $n \geq \binom{t+1}{2} > t^2/2$, and the desired result follows.

In [22], it has been shown that for (n, m, t)-CFF with $t \geq 2$, $m \geq c\frac{t^2}{\log t} \log n$ for some constant c which is approximately $1/2$. On the other hand, Erdös *et al* [12] showed that for any $n > 0$, there exists an (n, m, t)-CFF with $m = O(t^2 \log n)$ and $|B_i| = O(t \log n)$. This result is, however, non-constructive. Although Kumar *et al* [16] gave a probabilistic construction of CFF that meets the bound, explicit constructions that can achieve or get close to Erdös *et al* bounds are still of interest.

We are now discussing SCFF with $\ell = v$. This will imply that we can construct a t-out-of-n secret sharing scheme from a v-out-of-v scheme.

Theorem 13. *Constructing a (t, n) scheme from a (v, v) scheme using a (v, n, t)-SCFF is only possible if $v \geq \binom{n}{t-1}$.*

Proof. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the n participants in the considered (t, n) secret sharing scheme, $\Gamma_{t-1} = \{A \mid A \subseteq \mathcal{P}, |A| = t-1\}$. Let $(X, \mathcal{B})$ be the (v, n, t)-SCFF, where P_i is assigned a block $B_i \in \mathcal{B}$. We define a function that associates to a set A in Γ_{t-1} a subset of X that are not the union of the blocks allocated to A, we then prove that any different elements in Γ_{t-1} are mapped to disjoint subset of X. The result then follows directly.

The above claimed function is defined as

$$g : \Gamma_{t-1} \to 2^X,$$

such that for any $A \in \Gamma_{t-1}$, where $A = \{P_{i_1}, \ldots, P_{i_{t-1}}\}$ then

$$g(A) = X \setminus \cup_{j=1}^{t-1} B_{i_j}.$$

For any $A_1, A_2 \in \Gamma_{t-1}$, if $g(A_1) \cap g(A_2) \neq \emptyset$, then there is an element $x_i \in X$ which is in both $g(A_1)$ and $g(A_2)$. So x_i can not be covered by the union of blocks allocated the participants in A_1 and A_2. This contradicts the assumption that the union of t blocks cover X since $|A_1 \cup A_2| \geq t$.

We give a construction to show that the bound in Theorem 13 is tight when $t > 1$. Let $\Gamma_{t-1} = \{A \mid A \subseteq \mathcal{P}, |A| = t-1\}$. Let X be a $\binom{n}{t-1}$-set indexed by the elements in Γ_{t-1}, that is $X = \{X_A \mid A \in \Gamma_{t-1}\}$. For each $1 \leq i \leq n$, we define $B_i = \{X_B \mid P_i \notin B, B \in \Gamma_{t-1}\}$. Let $\mathcal{B} = \{B_1, \ldots, B_n\}$. Then it is straightforward to verify that $(X, \mathcal{B})$ is a $(\binom{n}{t-1}, n, t)$-SCFF, and the claim for the tight bound follows. Note that the case $t = 2$ was already proposed in [7] to obtain a homomorphic secret sharing scheme.

6 Evaluation and Conclusions

In the following we translate some of the results in the previous section into constructions of mechanical threshold schemes from old ones. We compute possible

values of ℓ which are not necessarily maximal or minimal (see the discussion in 4). Here some possible parameters:

1. For any integer $v \equiv 1, 4 \bmod 12$, an (ℓ, v) scheme results in a $(3, \frac{v(v-1)}{12})$ scheme (Corollary 1). The value $\ell = 9$ is possible for these schemes;
2. For any prime power q and any integer $\mu < q$, a $(\ell, q^2 + q)$ scheme results in (t, q^μ) scheme provided $\binom{t}{2} < \frac{q-1}{\mu-1}$ (Corollary 2). For these, (at least) the following values of ℓ are possible: $(t-1)*(q+1)+1 \leq \ell \leq t*(q+1)-\binom{t}{2}*(\mu-1)$.
3. For any prime power q and integer $i \geq 1$, a (ℓ, q^{i+1}) scheme results in a (t, q^{2^i}) scheme provided $t \leq \sqrt{\frac{2q}{i}} + 1$ (Corollary 4). For these, (at least) the following values of ℓ are possible: $(t-1) * q^i + 1 \leq \ell \leq t * q^i - \binom{t}{2} * i * q^{i-1}$.
4. For any prime power q and even m, a (ℓ, q^m) scheme results in a $(t, q^{cmq^{m/2}})$ scheme provided $q > 2\binom{t}{2}$, where c is some fixed constant (Theorem 6). For these, (at least) the following values of ℓ are possible: $(t-1) * q^m + 1 \leq \ell \leq t * q^m - \binom{t}{2} * (q^{m-1} + q^{m/2-1}\sqrt{q^m})$.

For example using $q = 4$, $\mu = 2$ and the parameters in 2 we can construct a $(2, 16)$ scheme from a $(\ell, 20)$ scheme where $6 \leq \ell \leq 9$. Using the same value of q and $i = 1$ together with the result in 3 we obtain a $(3, 16)$ scheme from a $(9, 16)$ and also a $(2, 16)$ scheme from a $(\ell, 16)$ where $5 \leq \ell \leq 7$.

We point out that constructing new threshold schemes from old ones using SCFF may have other applications such as constructing multiplicative secret sharing schemes which are of high importance in threshold cryptography. It is easy to show that if the old scheme is multiplicative then the new scheme based on SCFF is also multiplicative, and its asymptotic efficiency (share expansion) matches the best known results in [5]. However it is not clear if the SCFF construction can outperform the recursive construction in [5] in other aspects. Comparing the two approaches in constructing multiplicative secret sharing schemes deserves a more careful treatment.

References

1. F. Bao, R. Deng, Y. Han, and A. Jeng. Design and analysis of two basic protocols for use in TTP-based key escrow. *Information Security and Privacy, Second Australian Conference, ACISP '97*, LNCS **1270** (1997), 261–270. Sydney, NSW, Australia, July 7–9.
2. J. C. Benaloh, Secret sharing homomorphisms: Keeping shares of a secret secret, *Advances in Cryptology–Crypto '86*, LNCS, **263**(1986), 251–260.
3. J. Bierbrauer, T. Johansson, G. Kabatianskii and B. Smeets, On families of hash functions via geometric codes and concatenation, *Advances in Cryptology–CRYPTO'93*, LNCS, **773** (1994), 331–342.
4. G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.
5. S. R. Blackburn, M. Burmester, Y. Desmedt and P. R. Wild, 'Efficient multiplicative sharing schemes,' in *Advance in Cryptology–Eurocrypt '96*, LNCS, **1070**(1996), 107-118.

6. P. J. Cameron and J. H. Van Lint, Deisigns, Graphs, Codes, and their Links, Cambridge University Press, Cambridge 1991.
7. C. Boyd. Digital multisignatures. In H. Beker and F. Piper, editors, *Cryptography and coding*, pp. 241–246. Clarendon Press, 1989. Royal Agricultural College, Cirencester, December 15–17, 1986.
8. J. L. Carter and M. N. Wegman, Universal classes of hash functions, *Journal of Computer and System Sci.*, **18**(1979), 143–154
9. L. Chen, D. Gollmann, and C. Mitchell. Key escrow in mutually mistrusting domains. *Security Protocols* LNCS **1189** (1997), 139–153.
10. Y. Desmedt and S. Jajodia, Redistributing secret shares to new access structures and its applications, *Preprint*, 1997.
11. J. Diamant and F. Rhee. FBI follows money to 7 close to home, catches Ghantt in Mexico. The Charlotte Observer, March 3, 1998. See also: `http://www.charlotte.com/observer/special/heist/pub/heist.htm`.
12. P. Erdös, P. Frankl, and Z. Furedi, Families of finite sets in which no sets is covered by the union of r others, *Israel Journal of Mathematics*, **51**(1985), 79–89.
13. Y. Frankel and P. Gemmell and P. D. MacKenzie and M. Yung, Optimal Resilience Proactive Public Key Cryptosystems, *38th Annual Symp. on Foundations of Computer Science (FOCS)*, 1997.
14. T. Helleseth and T. Johansson, Universal hash functions from exponential sums over finite fields and Galois Rings, *Advances in Cryptology–Crypto'96*, LNCS, **1109**(1996), 31–44.
15. J. W. P. Hirschfeld. *Projective Geometries over finite fields.* Oxford University Press, N.Y., 1979.
16. R. Kumar, S. Rajagopalan and A. Sahai. Coding constructions for blacklisting problems without computational assumptions, *Advances in Cryptology – CRYPTO '99*, LNCS, **1666**(1999), 609-623.
17. K. Martin, R. Safavi-Naini, and H. Wang, Bounds and techniques for efficient redistribution of secret shares to new access structures, *The Computer Journal*, **42**(8) (1999), 638–649.
18. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *Proceedings of the twenty first annual ACM Symp. Theory of Computing, STOC*, (1989), 33–43.
19. A. Shamir, How to Share a Secret, *Communications of the ACM*, **22**(1976), 612–613.
20. D. R. Stinson, Universal hashing and authentication codes, *Advances in Cryptology–CRYPTO '91*, LNCS, **576** (1992), 74–85.
21. D. R. Stinson and R. Wei, Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes, *SIAM. J. Discrete Math*, **11**(1998)41–53.
22. D. S. Stinson, R. Wei and L. Zhu, Some new bounds for cover-free families, *Journal of Combinatorial Theory, A*, **90**(2000), 224-234.
23. J. H. van Lint, Introduction to Coding Theory, *Graduate Texts in Mathematics*, Vol. **86**, Springer-Verlag, 1992.
24. M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, **22** (1981), 265–279.
25. F. J. MacWilliams and N. J. Sloane. *The Theory of Error-Correcting Codes.* northholland publishing company, 1978.

Reliable MIX Cascade Networks through Reputation

Roger Dingledine[1] and Paul Syverson[2]

[1] The Free Haven Project
arma@mit.edu
[2] Naval Research Lab
syverson@itd.nrl.navy.mil

Abstract. We describe a MIX cascade protocol and a reputation system that together increase the reliability of a network of MIX cascades. In our protocol, MIX nodes periodically generate a communally random seed that, along with their reputations, determines cascade configuration. Nodes send test messages to monitor their cascades. Senders can also demonstrate message decryptions to convince honest cascade members that a cascade is misbehaving. By allowing any node to declare the failure of its own cascade, we eliminate the need for global trusted witnesses.

Keywords: anonymity, reputation, peer-to-peer, communal randomness

1 Introduction

Practical anonymous communication systems require high reliability. Reliability can lead to efficiency because routes are more likely to succeed. Reliability can also improve anonymity because senders need to resend fewer messages, and because a reliable system draws more users and thereby increases anonymity sets. Past approaches to increasing remailer reliability have included writing more reliable software [26], building MIX protocols that give provable robustness guarantees [6,13,21], and building a reputation system to let users choose paths based on the published scores for each node [7].

The reputation system described in [7] uses a MIX-net in which nodes give receipts for intermediate messages. These receipts, together with a set of witnesses, allow senders to verify the correctness of each node and prove misbehavior to the witnesses. Here we investigate and solve two problems from that design:

- *The mechanism for verifying a failure claim requires a set of global witnesses, a threshold of which need to be involved in confirming every failure claim. These global witnesses create both a trust bottleneck and a communications bottleneck.* Our new reputation system avoids these bottlenecks by making each node a witness to its own cascade.
- *An adversary trying to do traffic analysis can get more traffic by gaining a high reputation.* We protect against such adversaries by choosing from a pool of "acceptable" MIX nodes, and building cascades so we can bound the probability that an adversary will control an entire cascade.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 253–268, 2003.

2 Overview and Threat Model

We aim to improve the reliability of anonymous communication systems, enabling their practical use in fields such as electronic commerce, where transactions need to be efficient, reliable — and, frequently, anonymous. We order MIX nodes into cascades, where each cascade presents a fixed path through the network. This approach allows us to decentralize the use of witnesses to detect failure as introduced in [7], since each node can witness the traffic through its own cascade. Further, using cascades allows us to better resist traffic analysis from a pervasive adversary, since sending traffic through fixed paths makes intersection attacks more difficult. Our reputation system gives users a more accurate picture of which nodes are currently up, allowing them to choose routes more reliably and to know what level of protection they're getting.

Specifically, we aim to defend against two adversary goals. An *anonymity-breaking* adversary tries to discover linkability between sender and receiver; to identify the sender or receiver of a given message; or to trace a sender forward (or a receiver backward) to any messages. A *reliability-breaking* adversary tries to deny service to users. We assume that our adversary can passively watch all traffic, and can delay, modify, or insert some messages. We also assume that the adversary has compromised some fraction of the participating MIXes.

Section 4 shows the protocol by which MIXes build themselves into cascades in a public and verifiable way, and also describes the process of generating *communal randomness* so MIXes don't need to trust a central authority to build the cascades. We periodically rebuild cascades to reflect changes in reliability.

Inspired by a comment by Jim McCoy about using reputation capital to regulate participants in DC-nets [20], we use a very simple reputation system. Any member of a cascade can declare its own cascade to have failed. Nodes in a successful cascade each gain one reputation point, whereas all nodes in a failed cascade lose one point. Nodes that misbehave by incorrectly reporting cascade failure thus damage their own reputations too. We describe this reputation system in Section 5, including some proposals for limiting the fraction of adversary-controlled nodes, a discussion of some pitfalls introduced by our simple reputation system, and our technique for building cascades to reduce the chance that an adversary will control an entire cascade.

Section 6 describes our modified MIX cascade protocol, and also specifies how MIXes can decide when their cascade has failed. In Section 7, we describe a variety of attacks on the system and examine how well our design withstands these attacks. Finally, we close in Section 8 with a discussion of possible directions for future research.

3 Related Work

3.1 MIX-Nets

Chaum introduced the concept of a MIX-net for anonymous communications [5]. A MIX-net consists of a group of servers, called MIXes (or MIX nodes), each of

which is associated with a public key. Each MIX receives encrypted messages, which are then decrypted, batched, reordered, stripped of the sender's name and identifying information, and forwarded on. Chaum also proved security of MIXes against a *passive adversary* who can eavesdrop on all communications between MIXes but is unable to observe the reordering inside each MIX.

One type of MIX hierarchy is a cascade. In a cascade network, users choose from a set of fixed paths through the MIX-net. Cascades can provide greater anonymity against a large adversary, because in a free-route system an adversary who owns many of the MIXes can use intersection attacks to dramatically reduce the set of possible senders or receivers for a given message [3].

Current research on MIX-nets includes stop-and-go MIX-nets [16], distributed flash MIXes [13], and hybrid MIXes [23]. MIX cascade research includes real-time MIXes [15] and web MIXes [2].

3.2 Robustness and Reliability in MIX-Nets

Previous work primarily investigates the *robustness* of MIX-nets in the context of a distributed MIX system [13]. A MIX is considered robust if it survives the failure of any k of n participating servers, for some threshold k. This robustness is all-or-nothing: either k servers are good and the MIX works, or they are not good and the MIX likely will not work.

Robustness has been achieved primarily via zero-knowledge proofs of correct computation. Jakobsson showed how to use precomputation to reduce the overhead of such a MIX network to about 160 modular multiplications per message per server [13], but the protocol was later found to be flawed [21] by Mitomo and Kurosawa. Desmedt and Kurosawa's alternate approach [6] requires many participating servers. Abe's MIX [1] provides *universal verifiability* in which any observer can determine after the fact whether a MIX cheated, but the protocol is still computationally expensive. Neff recently made further efficiency improvements to universally verifiable mixing [22].

Reliability differs from robustness in that we do not try to ensure that messages are delivered even when some nodes fail. Dingledine et al's reputation system for free-route MIX-net reliability [7] aims instead to improve a sender's long-term odds of choosing a MIX path that avoids failing nodes. This work similarly attempts to provide more reliable long-term service by identifying reliable MIXes and building cascades from them.

We note that reliability and robustness can be composed: a cascade or distributed MIX with robustness guarantees can be considered as a single node with its own reputation in a larger MIX-net.

3.3 Approaches to MIX-Net Reliability

MIX-net protocols can give specific guarantees of robustness. Under suitably specified adversary models, these results may be quite strong, e.g. "this distributed MIX delivers correctly if no more than half of its participating servers are corrupt." Such protocols are often complicated and inefficient.

Levien's statistics pages [19] present another approach. They track both remailer capabilities (such as what kinds of encryption the remailer supports) and remailer up-times, observed by pinging the machines in question and by sending test messages through each machine or group of machines. Such *reputation systems* improve the reliability of MIX-nets by allowing users to avoid choosing unreliable MIXes.

Instead of engineering the MIX-net protocol directly to provide reliability, we make use of reputations to track MIX performance. In this approach, we specify a set of behaviors that characterize a functioning or failed MIX. We are not likely to prove strong theorems — the goal of reputation is to make the system "better" without guaranteeing perfection. Like Levien's reputation system for free-route MIX networks, our published reputations enable users to find better routes. Further, our reputation system works behind the scenes to build more reliable cascades — a process that may even be entirely transparent to the users.

Reliability via protocol is the most well-studied approach, while reliability via reputations in the form of Levien statistics is the most widely used. Our work combines the two approaches: we modify the MIX-net protocol to support easier detection of MIX failures and then specify a suitable reputation system.

4 How to Randomly Self-Build Cascades

We periodically rearrange the nodes into cascades, so that cascades reflect recent changes in reliability, and so nodes in failed cascades can get back in a working cascade. We choose to rearrange *all* nodes, not just those from failed cascades; otherwise reliable nodes will concentrate in stable cascades and unreliable nodes in unstable ones, making it difficult for new good nodes to gain reputation.

Cascades rebuild with period T (e.g., one day). By $T-a-b$, each participant sends a sealed commitment to the Configuration Server (CS). At $T-a-b$, the CS publishes the set of commitments. By $T-b$, participants should reveal to the CS. At T, the CS publishes the set of reveals, along with the configuration of cascades for that round. Observers can verify that the CS followed the configuration scheme and used the contribution from each node.

The CS gives each node N a signed receipt $\texttt{sign}(CS, [\textit{commitment}, \textit{timestamp}])$. The period from $T-a-b$ to $T-b$ should be long enough that anyone within reasonable clock skew can verify that the commitment phase has truly closed. Adequate time must be allowed for nodes to submit their secrets and to make use of a certified delivery service if their submissions are not accepted and receipts provided by the CS in a timely manner. The CS also provides a receipt for the reveal. With both receipts, N can prove that he should be included in the next cascade configuration.

Commitment from N: $\texttt{sign}(N, [N, \texttt{IP}, \texttt{port}, \texttt{bandwidthpledge}, \texttt{tsbc}(\texttt{rand}_N)])$.

Reveal from N: $\texttt{sign}(N, [N, \texttt{IP}, \texttt{port}, \texttt{bandwidthpledge}, \texttt{rand}_N])$

Here "$\texttt{tsbc}(\texttt{rand}_N)$" is N's temporarily-secret bit-commitment to his random value (to be explained below, in Section 4.1). The CS then builds a new set of cascades for that T, arranged according to an unpredictable value communally generated by the participating mixes. Thus, no one can control the configuration of cascades. Bandwidth might be divided into four categories: 10Kb/s,

100Kb/s, 1Mb/s, 10Mb/s. Participants choose exactly one of these values for their bandwidth pledge, and each of the four buckets is formed into a set of cascades according to the algorithm described in Section 5, using the unpredictable communal value.

The communal value can be used as a seed to a PRNG, so the amount of randomness required from each participant is quite small. If N is worried that CS will ignore his commitment or reveal (not provide a receipt), he can use a certified mail delivery system [24,28] to convince other people that CS is misbehaving. Alternatively, we can build our own certified delivery system by pre-assigning a set of witnesses (perhaps fifteen) from the previous T. A threshold of these witnesses is sufficient to prove misbehavior. As long as the adversary does not control too high a percentage of nodes then we can trust this system (cf., Section 5 for discussion of how we limit this).

The set of cascades is determined by the communal value that is publicly verifiably committed when the CS signs and posts the commitments and their revealed values. Because a node could still control the resulting communal value by choosing whether to reveal its value, the commitment is done such that if it is not revealed, the CS can uncover it after a predictable amount of computation. Any committed value that is not revealed by $T - b$ should be computed by the CS to be revealed at T.

4.1 Communal Randomness via Temporarily Secret Commitments

Communal randomness, or more precisely a communally determined unpredictable value, is obtained by collecting random values from participating MIXes. These are kept secret until everyone has committed so that no one can predict the result of combining them. The committed values can be uncovered without help from the committers if necessary so that no one can alter the communal result in a predictable way by failing to reveal what she committed. On the other hand, not all the committed random values can be uncovered by an adversary in time for him to craft a commitment that will yield a predictable result.

Various approaches to this capability have been published. In [11], the committed unpredictability comes from a long "delaying" calculation on the result of a (fast) combination of the inputs from participants. Another similar scheme is given in [27]. Both [4] and [27] give commitment schemes that are individual, as here; [4] also describes a zero-knowledge proof of a well-formed commitment. [12] further expands and develops the work in [11] and [27].

We follow the individual commitment approach of [12,27] because it is both shortcut computable and easily commitment-verifiable. Anyone possessing a secret (in this case the committer) can compute the committed result quickly. Once the committed value is revealed, anyone can easily check that this is the committed value. Commitments take the form $\mathtt{tsbc}(\mathtt{rand}_N) = \langle \mathtt{enc}(K, \mathtt{rand}_N), w(K) \rangle$, the encryption of $\mathtt{rand}_N$ with K followed by a TSBC function used to commit to K. [25] presents a function such that the committer N can quickly and easily calculate $w(K)$ as well as $\mathtt{enc}(K, \mathtt{rand}_N)$. Once K has been revealed (or calculated) for each of the entries, all the keys can be published by the CS.

Anyone can quickly verify the communal unpredictable value by performing the encryptions and computing their amalgamation.

Neither shortcut computability nor easy commitment-verifiability hold for the collective commitment that first appeared in [11]. The application of commitments to communal unpredictability is not the central focus of [4] and is only described briefly. It is also only described for two parties producing a random bit; thus only one commitment from one party is needed to run the protocol. With multiple parties committing to many bits, malleability of the timed-commitments [9] may become a factor if the revealed values are simply XORed together to produce the result. The well-formedness proofs may ultimately play a role in the non-malleability of the commitments since this is similar to how non-malleability of commitments is proved; we leave this as future work. Instead, we avoid the issue of malleability by ensuring that the amalgamation of the revealed values is not affected by correlations among those revealed values. For example, if the revealed values are concatenated in the order the commitments were posted and then hashed with a well-designed hash function, the result should be unpredictable if even one of the committed values is unpredictable.

We could make use of the well-formedness proofs in [4] to prevent nodes from submitting malformed commitments without detection, but the computational overhead is not necessary. Whether a commitment is malformed can be confirmed after a predictable amount of computation. Once it is known to be malformed, the result can either be discarded or it can be used as if it had been well-formed (details in [12,27]). Because inputs come from nodes with a vested interest in maintaining reputation, we can use the reputation system to ensure that malformed commitments are rare. All nodes must commit to participate in the network, but to minimize the number of malformed inputs or committed but not overtly revealed inputs, only commitments from high reputation nodes will contribute to the communal unpredictable value. We might use inputs from the top half of the nodes (all bandwidths), to make sure we have enough inputs to make collusion or compromise of the result unlikely. Any node that commits but does not reveal or puts in a malformed commitment loses reputation. To minimize ongoing problems from a misbehaving node, the random input from that node is not used for a fixed number of subsequent rounds.

If all nodes contributing to the communal value have revealed secret values that can be verified as committed during the entry phase, these can be combined by whatever means we use to yield a random result, e.g., concatenation and hashing as above. If not, we use the delaying calculation to uncover those not revealed. If all such revealed values correspond to well-formed TSBC entries, the result still remains easily commitment verifiable. If any of them are malformed, the communally determined value will only be verifiable by the corresponding delaying calculation (with parallel or probabilistic speedup, once it has been done initially, cf. [12,27] for details).

5 Reputation System and Cascade Configuration

We would like to develop as simple a scoring system as possible — simple systems are easier to analyze for security, and they allow the users to more easily

understand the implications of a given score. Our system decrements the reputation of all nodes in a failed cascade, and increments the reputation of all nodes in a successful cascade. Thus we don't have to worry about pinpointing the cause of failure. The hope is that reputations will give a general sense of the reliability of nodes over the long term.

However, because a node's behavior affects the reputation of its cascademates, this introduces a new attack on the reputation system. When a cascade fails that has fewer bad nodes than good nodes, it does more damage to the overall reputation of good nodes than bad. Since bad nodes can intentionally fail a cascade, they can exploit this vulnerability to gradually reduce the relative reputation of the good nodes.[1] The bad nodes "creep upward" on the reputation spectrum, eroding the reputation of nodes above them — our visual simulations led us to refer to this attack as the *creeping death.*

Because the bad nodes can gain reputation faster, they can eventually get to any point on the reputation scale. Rather than complicating the reputation system to prevent our adversary from performing the creeping death attack, we intentionally keep it simple and develop an algorithm for building cascades that minimizes impact from this attack.

Since bad nodes can position themselves anywhere (reputation-wise) to increase the chance of winning a whole cascade, the optimal strategy to protect anonymity chooses nodes for each cascade entirely at random. But we also want to increase the cost of reliability breaking, so that the adversary can only affect cascades likely to be highly desired for use if he runs reliable nodes himself; simply getting nodes into the system should have less impact. We thus combine these approaches and begin choosing cascade nodes randomly (using the random seed from the method of Section 4) from an adequately large set of nodes, but still of the highest possible reputation. (See Section 5.1 for details.)

While the reputation system reduces the impact of an adversary that merely gets nodes accepted into the system, the creeping death attack allows a resourceful adversary to quickly move up on the reputation spectrum. An adversary with many nodes can still succeed at breaking reliability and anonymity. We prevent our adversary from creating a multitude of identities and flooding the system with them (an attack known as *pseudospoofing*) with an identity-based barrier to entry: a web of trust like Advogato [17]. A web of trust allows us to limit the number of nodes an adversary can get certified. [17] gives a proof that the number of bad nodes accepted by the web is limited by the number of honest members that might assign trust to the adversary (*confused* nodes) — not the number of nodes the adversary creates. If we pick the seeds of the web carefully and make some assumptions about the number and position of confused nodes, we can bound the fraction of bad nodes in the system.

Alice should certify Bob based on whether she believes him to be trustworthy (to be a real person with good intentions). If she certifies based on expected performance, the adversary can simply run convincingly reliable nodes. Certifi-

[1] 'Good' and 'bad' refer to nodes that are honest or that are part of the adversary, respectively. Because a bad node may be reliable to break anonymity or reliability, they do not necessarily correspond to 'reliable' and 'unreliable'.

cation aims only to bound the total percentage of adversary-owned nodes in the system.

On the other hand, we might ask Alice to not certify Bob if she believes he might be unreliable. However, this imposes a greater burden on Alice, and also doesn't account for the fact that Bob's behavior can change over time (whereas certification is a one-time event). Instead, the reliability of an admitted node will be determined by the reputation system. Nodes that are the most reliable over time will have the highest reputations.

5.1 Building Cascades

It is tempting to believe that some alternative reputation system or cascade algorithm can reduce or simplify the problem introduced by creeping death. For instance, we might punish nodes incrementally more as they have more failures on record. But this is exactly the problem — honest nodes *do* fail more often than adversary nodes. For any pattern that we look for, our adversary can arrange it so honest nodes fit that pattern better or more often than bad nodes.

Consider cascades with only two nodes. In cascades with one good and one bad, the bad node can hurt only one good node, so by construction bad nodes do equal damage to good nodes. But even this system falls prey to the creeping death. Since a cascade can only fail if one of its nodes writes the "We failed" certificate, the both-bad case will *never* fail (the nodes simply never write the certificate), whereas a both-good case will sometimes legitimately fail. Thus every both-bad cascade can stop functioning immediately, yet the bad reputations increase faster than the good reputations over time.

While we cannot easily reduce or prevent creeping death, we can choose cascades to produce acceptable and predictable risk and reliability despite creeping death. Reasonable anonymity protection may require chaining cascades into longer paths.

We order the nodes by reputation, and choose nodes for the first cascade randomly from within a pool of nodes at the top of the reputation spectrum. (Randomness is obtained from the seed chosen in the method of Section 4.1.) Next, add to the pool enough next-highest reputation nodes to maintain its size and pick another cascade at random. This continues until the last cascade for which an adequate pool size can be maintained. At that point, the remaining nodes are formed into cascades at random.

How do we decide this pool size? Assume the following notation:

- p = fraction of nodes that are bad,
- s = scare factor: acceptable probability of adversary-controlled path,
- r = range: size of the pool from which nodes are chosen for a single cascade,
- l = length of a single cascade,
- c = chain length: number of cascades chained together,

For determining the range, we assume a worst case for adversary distribution, because of creeping death. That is, we always assume the adversary resides

entirely in the pool of nodes from which we're picking a cascade. This means

$$\left(\frac{p}{r}\right)^{lc} = s \quad \text{and so} \quad r = \frac{p}{s^{\frac{1}{lc}}}$$

For example, suppose that cascades are of length 4, there are not more than 20% bad nodes altogether, and it is acceptable that one of every hundred thousand paths (cascade chains) is completely bad — meaning messages through it are compromised. Also assume that we chain three cascades to reduce the odds that all nodes traversed are bad.[2] Then

$$r = \frac{.2}{(10^{-5})^{\frac{1}{12}}} = 0.522$$

Thus the first cascade must come from nodes in the top 52.2% of the reputation spectrum. The next cascade must be chosen from the same pool, minus the four nodes of the first cascade and plus the four next highest reputation nodes. Once the lowest reputation node has been added to the choice pool, the remaining nodes are just chosen at random until all the cascades are formed.

A chained cascade path is the same as a single long cascade of length lc with respect to the odds that all nodes in it are bad, but they are not the same in general. When a chained cascade fails, only the offending subcascade is removed from the system for that period and its nodes decremented in reputation. Also, users may choose to chain cascades or not, may not always choose to chain in the same way, or may choose a longer or shorter chain. A user may choose not to chain because of the computational overhead, the latency, etc. This choice will afford him improvements in those areas, but at an increased risk to anonymity. Users should be made aware of the risks. Our system allows an easy explicit presentation of the relative risks and tradeoffs. It also allows us to adjust these at the system level. For example, if we wish to reduce r, we can weaken s, or adjust l or the recommended c. Of course if p is high enough, s strong enough, and we limit lc to some practical bound, r may exceed 1 and no network is feasible. Or r may be close enough to 1 to render the reputation system largely moot — but at least we can calculate this and react accordingly.

6 The Cascade Protocol, or, When to Fail Your Cascade

Opportunies for misbehavior in cascades fall into three classes:

1. Entry point: Incoming messages might not be accepted.
2. Inside the cascade: Messages might be replaced with dummy messages.
3. Exit point: Messages might not be delivered.

Each MIX can test its cascade by sending and receiving messages using ordinary-looking external addresses — but spoofing or maintaining plausible

[2] A path length of 12 would be absurd with current remailers; but the whole point of this design is to improve reliability so long paths are feasible.

external addresses is hard. Instead, we protect against this "selectively process the test messages" attack by relaying traffic through other nodes in the cascade and allowing them to undetectably insert test messages. All l nodes in the cascade (typically l might be 4 or 5) accept $\frac{1}{l}$ of the total traffic, and deliver the messages to the head of the cascade. The head publishes a snapshot of the batch (a set of hashes of each message) as he processes it.

A sender Alice can ask for the snapshot to verify that her message got into the batch. If not, she concludes that either the head or the node she used was dishonest, and goes to a different node or cascade. As an optimization, nodes that accept messages can give Alice a receipt if they accept her message. If her message does not make it into the batch, Alice can broadcast the message and the receipt to the other nodes in the cascade; an honest cascade member will determine that the receipt should have been honored and fail the cascade.

Because all the messages to the head come from other nodes in the cascade, these nodes can insert indistinguishable test messages into the batches. If a test message does not make it to the tail, its sender fails the cascade. Since other nodes can't tell which messages are test messages, dishonest nodes risk being caught if they replace even one message with a dummy message. Thus our protocol detects misbehavior at the entry point (Goal 1).

In the naive delivery design, the tail delivers messages and also broadcasts them to the other nodes in the cascade. Every node attempts delivery. Since the tail can't tell who wrote a test message, he must deliver every message to every node in the cascade or risk failing the cascade. To prevent the tail from selectively dropping messages based on destination, nodes address some of their test messages to previous recipients. Thus, the tail must deliver even to a user not known to be running a node. (This reliability increase must be balanced with possible spam abuse.) A more efficient design assumes a PKI which includes all recipients. In this case, we shortcut the need for broadcasting when the original delivery attempt produces a signed receipt; we discuss this more in Section 6.1. By delivering outgoing traffic to all cascade members, our protocol detects misbehavior at the exit point (Goal 3).

A dishonest head can publish a correct batch snapshot but replace its (or a conspirator's) portion of messages with dummy messages. Because it knows its portion contains no test messages, all of those messages will be undetectably lost. We solve this by supporting external test messages as well. Alice might become suspicious because the cascade accepts messages but doesn't deliver them; she can send a test message, wait a while, and then reveal to everybody how the message should have decrypted. If at least one node in the cascade is honest, he will agree that he didn't see the message and fail the cascade. Thus we detect misbehavior after the batch snapshot is published. (Goal 2).

Senders may want to chain cascades for stronger anonymity. To make chaining cascades more robust, nodes consider delivery to a second cascade as a special case. We can specify the entire cascade rather than a single node as the next hop in the chain. Each node from the first cascade chooses a random node in the second cascade and attempts delivery. Nodes may verify that their message is included in the next cascade's batch, and claim misbehavior if not. With this modification, the head of a cascade must be able to detect duplicates in the

batch; however, since all nodes must already detect duplicates to foil replays, this presents no extra burden.

Since senders exposing a faulty cascade have no reason to chain their test messages through another cascade, some nodes need to explicitly send external test messages to other cascades and verify their delivery.

Test messages using real addresses help foil time-based intersection attacks. In a standard MIX network, an adversary with information about what users are active at what times can quickly narrow down the set of suspects based on when traffic is seen. An active adversary works even faster by knocking out suspects until traffic stops. Because cascade nodes send messages too, an address might get mail at other times as well.

Users worried about profiling should send each message through a different cascade, so an adversary who owns a few cascades cannot read all messages. Users worried about message linkability should send all messages through one cascade: a single compromised cascade can reveal linkability.

A decentralized algorithm would allow users to keep a similar anonymity set across cascade reconfigurations, further blocking time-based intersection attacks. On the other hand, an adversary targetting a specific user benefits from this predictable behavior. We leave this as future work.

Plaintext messages are distinguishable and so are less reliable. Further, since test messages with convincing plaintext are hard to write, nodes are unlikely to address tests to a recipient without a known public key. Optionally, outside users could contribute convincing plaintext messages to be used as test messages by a node; that way it could send a plaintext test message to the recipient and verify its delivery.

6.1 Delivery Receipts

Message recipients can give the tail a receipt when he delivers a message. The tail first attempts to get a receipt, which he can use to prove that he delivered the message. If he does not get a receipt (e.g., because the destination address refuses to provide a receipt or does not exist), he broadcasts the message to the other members of the cascade, who try to deliver. In any case they now know that he followed the protocol.

An unhappy sender Alice can contact some cascade node N and claim "T didn't deliver my message", along with a demonstration of the remaining decryptions between N and T. If N remembers what he passed to $N+1$, Alice can show N what it should have looked like when it got to T.

If N has already heard from T about its attempt to deliver the message, he knows T is blameless. If not, he can query T for a receipt — if T has no receipt, the cascade failed (either the message never got to T, or he did not deliver it).

Delivery receipts detect misbehavior as long as one of the nodes in the cascade is honest. If the honest node is the tail T and the message makes it that far, then the message will be delivered (cases 1 and 2 handle if the message doesn't make it that far). If T is bad, either he delivers the message to an honest N and it gets delivered, or he does not and Alice can convince N that the cascade is misbehaving.

If a recipient is not configured to return a receipt, the delivery still gets through — in this case, the message gets broadcast to the other cascade members, and each of them attempts delivery. In a sense, the use of receipts is just a bandwidth optimization.

Most related work assumes public keys for recipients are known to all parties. Without this PKI, the exit node can forge a signed receipt from the recipient. But since we don't need to link keys to external identities, a sender Alice can include Bob's signature verification key in her message, allowing any cascade node to verify Bob's receipt in a decentralized fashion.

6.2 Capacity-Attacking Adversary

Since nodes can refuse incoming messages by falsely claiming to be full, the number of messages processed by a cascade is at least proportional to the number of honest nodes in that cascade. By inserting indistinguishable test messages into its own batches, each node verifies that the rest of the cascade is successfully decrypting and passing on that fraction of its bandwidth promise, as well as guaranteeing that the batch provides a minimum level of anonymity. Nodes should pad if they don't have a full batch fraction; by failing the cascade if the amount of traffic coming in from the previous hop is outside suitable thresholds, each node verifies that the rest of the cascade is spending (even if wasting) its entire bandwidth promise.

By not accepting any messages and then not delivering the corresponding dummy traffic, bad nodes can spend slightly less bandwidth than good nodes. But if those bad nodes are delivering messages for the honest cascade members, they are doing no more damage than if they simply had not signed up that period. Thus our design frustrates a *capacity-breaking adversary* (a special case of the reliability-breaking adversary).

6.3 Resource Management and Reputation Servers

Because the highest-reputation cascade cannot process all of the traffic, cascades publish available capacity information, including the expected wait or available quality of service for messages. Users compare reputation and available QoS from each cascade, thus balancing load across the cascades.

A group of redundant reputation servers (*RS*) serve current node state. Nodes give each *RS* hourly *heartbeat* updates, and deliver failure messages immediately. Because each node signs and timestamps each certificate, an *RS* can at most fail to provide the newest certificate. Each *RS* works with the others to ensure correct data, perhaps by successively signing certificate bundles. Senders download the entire bundle of certificates if it is small enough, else they must query through the MIX-net or query via Private Information Retrieval [18] to privately download a random subset. Otherwise, the adversary could use that information to aid intersection attacks.

Actual deployment is still a complex problem; we leave this as future work.

7 Attacks and Defenses

Attacks on Anonymity

Have enough nodes to own an entire cascade. By using a web of trust, building cascades from a large enough pool of reliable nodes, and suggesting a safe minimum chain length, we control the chance that this attack will succeed.

Gain high reputation to read more traffic. Similarly, our cascade building algorithm blocks this attack.

Replay attack, message delaying, etc. We rely on the standard defenses offered by MIX-net protocols.

Trickle attack. If one node in the cascade is honest, at least $\frac{1}{l}$ of the traffic will be legitimate in every batch.

Intersection attack. Using MIX cascades rather than free routes helps to defend against intersection attacks from very large adversaries [3]. By encouraging users to pad with dummy messages when not sending traffic, and to continue using similar anonymity sets across cascade configurations, we can further complicate intersection attacks. However, a complete solution to the intersection attack remains an open problem.

Influence cascade configuration externally. Our algorithm for generating communally random uncertainty resists individuals and groups, as detailed in Section 4.

Compromise the cascade configuration server. Because the output of the CS is publicly verifiable, incorrect behavior can be detected.

Knock down uncompromised cascades to get more traffic. While the adversary can knock down other cascades, the low chance of owning an entire path limits the success of this attack.

Attacks on Capacity and Reliability

Flood nodes with messages. If this becomes a problem, we can integrate a ticket service such as that in [2], limiting the number of messages a given identity can generate for each batch. Other solutions include proofs of work and other micropayment schemes [8,10,14].

Knock down many cascades. We assume that our adversary is not strong enough to knock down all or most of the cascades in the system. If he only knocks down some, service continues as normal.

Block commitments to the Configuration Server. If the adversary launches a denial of service attack against the CS, the participants can together use a decentralized algorithm to simulate the CS (all of its operations are public and verifiable).

Flood the CS with commits. We only consider commits from nodes which have been certified in the web of trust, and we only need to actually use those commitments from relatively high-reputation nodes.

Refuse commitments at the Configuration Server. Because we allow certified delivery to the CS, refusing commitments can be detected by any observer.

Refuse incoming messages as a cascade member. This attack can work to reduce capacity; but the node is still spending most of its pledged bandwidth on correctly processing messages from other nodes, as well as generating dummy traffic in place of the refused traffic. This attack is very inefficient.

Selectively process only test messages. Our protocol addresses this possibility in Section 6.

Attacks on Reputations

Beat the web of trust. The security of the web of trust is perhaps the most critical assumption of our system. If an adversary can get a lot of nodes certified without spending effort for each one, he can start widespread attacks on anonymity, reliability, capacity, and reputations. On the other hand, getting just a few extra nodes certified does not buy him much.

Internal selective DoS — creeping death. The adversary can use the creeping death attack (fail the cascade if good nodes lose more reputation than bad nodes) to gain any position in the reputation spectrum. Our cascade building algorithm makes this technique ineffective at breaking anonymity; further, it may prove very expensive in terms of resources to move a large number of nodes to the top of the reputation spectrum.

External selective DoS — knock down the high-reputation cascades. Our "all for one and one for all" approach to reputation makes selective denial of service even easier than in [7]. Previously, to knock down a reliable node you needed to successfully flood it or cause it to not process messages correctly. Now you simply have to locate and knock down the weakest member of its cascade. However, this vulnerability is acceptable: removing one cascade does not deny service to the system as a whole, and as above it does not get you much closer to breaking anonymity.

8 Future Directions

We have described a protocol for improving reliability of anonymous communication networks, based on a MIX cascade design and a simple reputation system. There are a number of directions for future research:

- Better approaches to generating convincing destinations for dummy traffic, or reducing the bandwidth overhead of the current approaches, would make the overall design more reasonable.
- Improved cascade configuration algorithms would allow us to provide stronger anonymity and reliability.
- More research on the scope and characteristics of the creeping death attack might give insight on how to defeat it.
- More analysis on the attack-resistance of our reputation system might yield stronger proofs or a better design. For instance, can we guarantee bounds on work performed by the adversary under various models?
- Adapting this design to free-route MIX networks would allow it to be tested with a wider deployment in the current remailer system.

Acknowledgements. We thank Daniel Barkalow and Nick Mathewson for intelligent discussions about the reputation systems and cascade configuration; Ian Goldberg for thoughts on cascade configuration; Geoff Goodell and Marc Waldman for edits; Paul van Oorschot for a helpful conversation about hash functions; and David Molnar for early discussions about the design and for help with Section 3.

References

1. Masayuki Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Advances in Cryptology - EUROCRYPT 1998, LNCS Vol. 1403*. Springer-Verlag, 1998.
2. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Designing Privacy Enhancing Technologies, LNCS Vol. 2009*, pages 115 – 129. Springer-Verlag, 2000.
3. Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In *Designing Privacy Enhancing Technologies, LNCS Vol. 2009*, pages 30 – 45. Springer-Verlag, 2000.
4. Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology – CRYPTO 2000, LNCS Vol. 1880*, pages 236–254. Springer-Verlag, 2000.
5. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1982.
6. Yvo Desmedt and Kaoru Kurosawa. How to break a practical MIX and design a new one. In *Advances in Cryptology - EUROCRYPT 2000, LNCS Vol. 1803*. Springer-Verlag, 2000.
7. Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. Proceedings of the Information Hiding Workshop 2001. Also available from <http://www.freehaven.net/papers.html>.
8. Roger Dingledine, Michael J. Freedman, and David Molnar. Accountability. In *Peer-to-peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly and Associates, 2001.
9. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *23rd ACM Symposium on the Theory of Computing (STOC)*, pages 542–552, 1991. Full version available from the authors.
10. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. Technical Report CS95-20, Weizmann Institute, 1995. A preliminary version appeared in *Crypto '92*, pp. 129–147.
11. David M. Goldschlag and Stuart G. Stubblebine. Publicly verifiable lotteries: Applications of delaying functions. In *Financial Cryptography, FC'98, LNCS Vol. 1465*, pages 214–226. Springer-Verlag, 1998.
12. David M. Goldschlag, Stuart G. Stubblebine, and Paul F. Syverson. Temporarily hidden bit commitment and lottery applications. Submitted for journal publication.
13. Markus Jakobsson. Flash Mixing. In *Principles of Distributed Computing - PODC '99*. ACM, 1999.
14. Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99)*. Kluwer, September 1999.

15. Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Real-Time Mixes: A bandwidth-efficient anonymity protocol. IEEE Journal on Selected Areas in Communications 1998.
16. D. Kesdogan, M. Egner, and T. Büschkes. Stop-and-go MIXes providing probabilistic anonymity in an open system. In *Information Hiding Workshop 1998, LNCS Vol. 1525*. Springer Verlag, 1998.
17. Raph Levien. Advogato's trust metric. <http://www.advogato.org/trust-metric.html>.
18. Tal Malkin. *Private Information Retrieval.* PhD thesis, MIT, 2000. see <http://www.toc.lcs.mit.edu/~tal/>.
19. Tim May. Description of Levien's pinging service. <http://www2.pro-ns.net/~crypto/chapter8.html>.
20. Jim McCoy. Re: DC-net implementation via reputation capital. <http://www.privacy.nb.ca/cryptography/archives/coderpunks/new/1998-10/0114.html>.
21. M. Mitomo and K. Kurosawa. Attack for Flash MIX. In *Advances in Cryptology - ASIACRYPT 2000, LNCS Vol. 1976*. Springer-Verlag, 2000.
22. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In P. Samarati, editor, *8th ACM Conference on Computer and Communications Security (CCS-8)*, pages 116–125. ACM Press, November 2001.
23. M. Ohkubo and M. Abe. A Length-Invariant Hybrid MIX. In *Advances in Cryptology - ASIACRYPT 2000, LNCS Vol. 1976*. Springer-Verlag, 2000.
24. James Riordan and Bruce Schneier. A certified e-mail protocol with no trusted third party. *13th Annual Computer Security Applications Conference*, December 1998.
25. Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. MIT LCS technical memo MIT/LCS/TR-684, February 1996.
26. RProcess. Selective denial of service attacks. <http://www.eff.org/pub/Privacy/Anonymity/1999_09_DoS_remail_vuln.html>.
27. Paul Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *Computer Security Foundations Workshop (CSFW11)*, pages 2–13, Rockport Massachusetts, June 1998. IEEE CS Press.
28. Zhou and Gollmann. Certified electronic mail. In *ESORICS: European Symposium on Research in Computer Security, LNCS Vol. 1146*. Springer-Verlag, 1996.

Offline Payments with Auditable Tracing

Dennis Kügler and Holger Vogt

Department of Computer Science*
Darmstadt University of Technology
D-64283 Darmstadt, Germany
{kuegler, hvogt}@cdc.informatik.tu-darmstadt.de

Abstract. Tracing is an important mechanism to prevent crimes in anonymous payment systems. However, it is also a threat to the customer's privacy as long as its application cannot be controlled. Relying solely on trusted third parties for tracing is inadequate, as there are no strong guarantees that deanonymizations are only applied legally.
A recent tracing concept is auditable tracing, where the customer has the power to control the deanonymization. With auditable tracing no trust is required, while it offers comparable tracing mechanisms.
We present the first off-line payment system with auditable tracing. Our payment system supports coin and owner tracing as well as self deanonymization in the case of blackmailing.

1 Introduction

Anonymous electronic payment systems based on blind signatures [Cha83] have been suggested to protect the privacy of customers. However, von Solms and Naccache [vSN92] have shown that anonymity may be misused for untraceable blackmailing of customers, which they called "perfect crime". Furthermore, anonymity may prevent investigations of money laundering and illegal purchases. Due to these anonymity related problems payment systems with revocable anonymity have been requested by governments and banks, and thus tracing methods have been invented, where the withdrawal and the deposit of coins can be linked by two complementary tracing mechanisms [SPC95]:

Coin tracing: The withdrawn coins of a suspicious or blackmailed customer are deanonymized so that the bank will recognize these coins at deposit.

Owner tracing: The coins deposited by a suspicious merchant are deanonymized so that the identity of their withdrawer is revealed.

Payment systems with trusted third parties (e.g. [CMS96, JY96, FTY96, DFTY97]) can provide both tracing mechanisms very effectively, but also have several shortcomings: The introduction of a trusted third party causes additional costs, which neither the bank nor the customer is willing to pay for. But even

* This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the PhD program (Graduiertenkolleg) "Enabling Technologies for Electronic Commerce" at Darmstadt University of Technology.

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 269–281, 2003.

worse, the achieved level of anonymity is uncertain, as any misuse of tracing by the trusted third party cannot be detected. Demanding a quorum of trusted third parties to cooperate for tracing only guarantees increasing costs, while conspiring trusted third parties may still be able to undetectably trace without permission.

Recent approaches for payment systems abandon the idea of trusted third party tracing [STS99, PS01, KV01a], but they only protect against blackmailing and lack support for coin and owner tracing. It was shown in [KV01b, KV01c] that coin and owner tracing also can be implemented without any trusted third party by introducing an audit concept. However, these payment systems require the bank to be *on-line* at payment.

In this paper we show that coin and owner tracing can also be implemented *off-line* without any trusted third party. Our payment system is based on [CMS96, CMS97], but replaces their trustee based tracing mechanism with the concept of *auditable tracing*: At a certain point of time customers can detect whether their spent coins have been traced or not and whether this tracing was performed with the permission of a judge or the customer himself. As every application of illegal tracing (i.e. tracing without permission) can be prosecuted, only legal tracing will be applied in practice. Thus, auditable tracing provides better protection of the customer's privacy than uncontrolled trusted third party based tracing.

Additionally, we support the self deanonymization mechanism of [PS01], which enables a blackmailed customer to always trace his blackmailed coins.

The remainder is structured as follows: In the next section we explain our idea for off-line auditable tracing. Section 3 presents the building blocks of our payment system, which is implemented in Section 4. The tracing mechanisms and the audit mechanism are described in Section 5. The general trade-off between tracing mechanisms and privacy is discussed in Section 6.

2 Offline Auditable Tracing

In payment systems with passive trustees (e.g. [CMS96, DFTY97]) a single public tracing key exists, for which only the trustee knows the corresponding private tracing key. At withdrawal the customer is forced to use the public tracing key to encrypt some information identifying this withdrawal. This information can only be decrypted by the trustee, which enables to link the payment and withdrawal view of this coin. Thus, coin and owner tracing can be performed.

Our payment system is not based on trusted third party tracing, but also uses tracing keys for encrypting identifying information. In contrast to payment systems with passive trustees a different public tracing key for each customer and merchant is issued by the bank. All tracing keys have a limited validity period and must be exchanged regularly against new keys. The application of tracing is restricted as follows:

- Coin tracing is possible, if the bank knows the private tracing key for the customer, who withdraws the coin.

- Owner tracing is possible, if the bank knows the private tracing key for the merchant, who deposits the coin.

Whether or not the bank knows a private tracing key depends on the *key generation parameter* for this tracing key. The bank reveals the key generation parameter, after a tracing key has expired. This enables the customer or the merchant to check and even prove whether they have been traced or not.

2.1 Legal Application of Tracing

When the bank creates a tracing key, it issues a *user certificate* containing an identifier for the user, the public tracing key, the activation date, the expiration date, the audit date, a transaction limit, and a symmetric encryption of the key generation parameter. We distinguish two types of user certificates:

- The *customer certificate* can only be used to withdraw coins until it becomes invalid.
- The *merchant certificate* can only be used to deposit coins until it becomes invalid.

Both types of user certificates (and thus the tracing keys) are valid from the activation date until the expiration date or until the transaction limit has been reached. The transaction limit additionally restricts the usage of a certificate e.g. to the maximum amount of money that can be withdrawn/deposited or to the maximum number of withdrawals/deposits.

When the audit date is reached, the customer or merchant can ask the bank for the symmetric key that was used to encrypt the key generation parameters in the certificate. Then the user knows whether the bank was able to trace or not.

To determine whether a detected tracing was legal or illegal, the user can request a *tracing certificate* from the bank. This tracing certificate has to be issued by a judge, who has permitted tracing this user. The tracing certificate is simply the user certificate signed by the judge.

2.2 Audit of Tracing Keys

Due to security and efficiency coin generations are necessary [Sch97] and every coin has only a limited life time. The bank issues coins during a *generate phase* and accepts them for deposit during an *accept phase*. After the bank stops issuing coins of a generation, it immediately starts issuing coins of the next generation.

The user certificates are related to coin generations as follows:

- **Customer certificates:** The activation and expiration dates of a customer certificate must be during the generate phase of the same coin generation. The audit date of the customer certificate is always a certain time period Ω after the end of the accept phase of this coin generation. As the coins are no longer valid for payments, it is possible to reveal at the audit date, whether coin tracing has been applied to these coins or not.

- **Merchant certificates:** The activation and expiration dates of a merchant certificate must be during the accept phase of the same coin generation. The audit date of the merchant certificate is always a certain time period Δ after the expiration date of the merchant's certificate. As this merchant certificate is no longer valid for deposits, it is possible to reveal at the audit date, whether owner tracing has been applied to these coins or not.

The relation of user certificates to coin generations is shown in Figure 1. Coin tracing is guaranteed to be undetectable for at least the time period Ω, and owner tracing is at least undetectable for the time period Δ. The time periods Ω and Δ can be used for investigations of suspected criminal activities. The length of Ω and Δ can even be chosen differently for different users, e.g. a previously convicted person may have later audit dates.

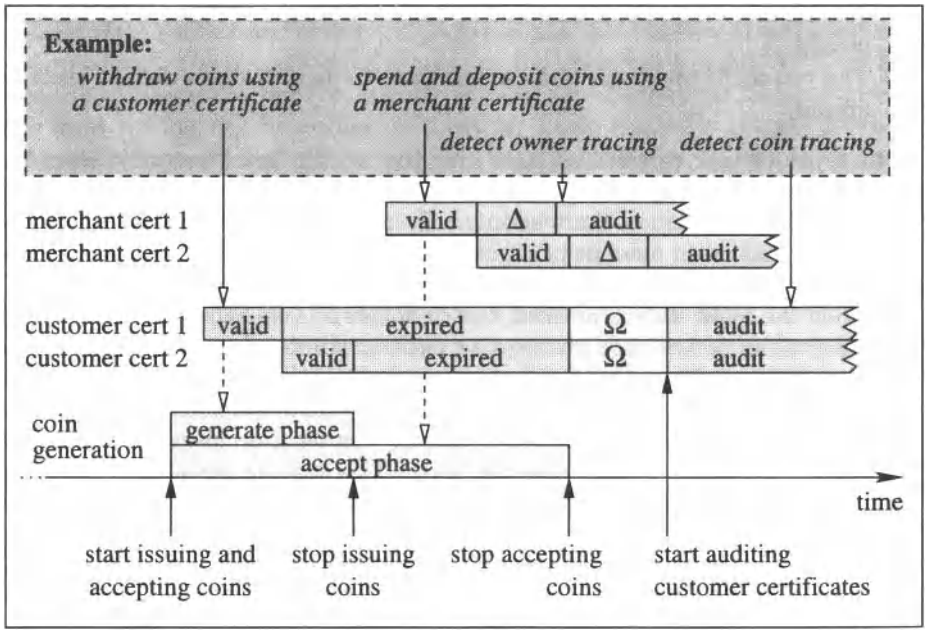


Fig. 1. The relation between tracing keys and a coin generation.

2.3 Tracing Blackmailers

The tracing mechanism described so far has the property that if a customer is traced, the bank alone executes the coin tracing at withdrawal. If the customer is not traced, nobody can execute coin tracing later. However, if the customer is blackmailed, coin tracing will be desired by the customer. Thus, he has to instruct the bank to apply coin tracing at withdrawal. The customer may use

a distress mechanism to inform the bank about the blackmailing, as described in [KV01a]. The bank will now create a public tracing key for which it knows the corresponding private key. As a blackmailer will demand a huge amount of money, the amount limit of the current customer certificate will be reached, and the bank can apply the new tracing key for coin tracing. Thus it is guaranteed that the bank can trace the majority of the blackmailed coins with the new tracing key. These coins can be invalidated by blacklisting and even refunded to the customer, if the blackmailer has not spent them yet.

While this is a solution to the blackmailing problem, it depends on the customer to inform the bank about the blackmailing *before* the coins are withdrawn. If the customer did not dare to notify the bank, all the coins withdrawn by the blackmailer will usually be untraceable. A simple protection against this is a policy issued and signed by the customer in advance. This customer policy is used to identify suspicious transactions and instructs the bank to trace those coins. An example for such a customer policy is the withdrawal of more than \$500 per day, which will trigger tracing with the next tracing key.

Another approach is a self deanonymization mechanism proposed by Pfitzmann and Sadeghi [PS01]. There only the customer is able to trace his own coins in the case of blackmailing. We adopt this idea in a way that it does not restrict the tracing capabilities of the bank. Our approach has the following properties: If the bank has performed coin tracing at withdrawal, it can trace alone. Otherwise, tracing is possible only if the customer assists the bank. This kind of tracing is available to the customer at any time so that deanonymization and blacklisting of coins are possible afterwards.

However, if the customer fears that he may be forced to deanonymize himself with this tracing method (e.g. due to laws that demand to reveal decryption keys like the British Regulation of Investigatory Powers Act), he can provably renounce the ability to trace himself.

The three mechanisms described above provide a very good choice for the customer to protect himself against blackmailing. Depending on his preferences he can choose one or more of these mechanisms when he opens his bank account. However, the blackmailer must not be able to alter the chosen mechanisms. To change the customer's preferences for those mechanisms e.g. the customer himself has to appear at the bank or alternatively the change should take effect after a certain time period.

Note that we omit bank robberies [JY96] as generic countermeasures have been proposed against this attack [KV01b], which can also be applied to this payment system.

3 Building Blocks

All computations are done in a cyclic group G of prime order q, where q is chosen so that the discrete logarithm problem is infeasible. For every coin generation the bank computes new parameters for the payment system. The three generators g, g_1, and g_2 are created by a pseudo-random function, which ensures that nobody knows the discrete logarithm between any of these generators. The bank chooses

a private signature key $x \in_R \mathbb{Z}_q$ and computes the public keys $y = g^x$, $y_1 = g_1^x$, and $y_2 = g_2^x$. Then G, g, g_1, g_2, y, y_1, and y_2 are published.

3.1 Merchant Tracing Keys

To implement owner tracing every merchant is assigned a public tracing key y_M. The merchant receives this key from the bank, who also provides the merchant certificate for this key. The corresponding private tracing key is the discrete logarithm of y_M to the base g_1.

The bank chooses a secret value $x_M \in_R \mathbb{Z}_q^*$ to compute the public tracing key y_M. If the merchant is traced, the bank computes $y_M = g_1^{x_M}$ so that x_M is the private tracing key. If no owner tracing is wanted, y_M is computed as g^{x_M}. This guarantees that the bank does not know the discrete logarithm of y_M to the base g_1.

The merchant does not know whether y_M was computed as $g_1^{x_M}$ or g^{x_M}. Thus, he cannot determine whether he is traced or not. The bank encrypts x_M in the merchant certificate and will reveal the symmetric encryption key only after the audit date.

3.2 Customer Tracing Keys

When the customer opens his bank account, he has to register a public key g_C. If he wants to use the self deanonymization mechanism, he must know the discrete logarithm of g_C to the base g_2. Otherwise self deanonymization is impossible.

The customer chooses a secret value $x_s \in_R \mathbb{Z}_q^*$ and either computes $g_C = g_2^{x_s}$ to enable self deanonymization or $g_C = g^{x_s}$, which later enables the proof that he does not know the discrete logarithm of g_C to the base g_2.

To implement coin tracing the bank assigns the customer a public tracing key y_C. The customer receives this key from the bank, who also provides the customer certificate for this key. The corresponding private tracing key is the discrete logarithm of y_C to the base g_2.

The bank chooses a secret value $x_C \in_R \mathbb{Z}_q^*$ to compute the public tracing key y_C. If the customer is traced, the bank computes $y_C = g_2^{x_C}$ so that x_C is the private tracing key. If no coin tracing is wanted, y_C is computed as $g_C^{x_C}$. This guarantees that the bank does not know the discrete logarithm of y_C to the base g_2. However, if the customer has chosen $g_C = g_2^{x_s}$, the bank and the customer together know the discrete logarithm and can apply coin tracing.

The bank encrypts x_C in the customer certificate using symmetric encryption. Only when the symmetric key and thus x_C is revealed after the audit date, the customer can determine how y_C was created and whether his coins are traced or not. If the bank encrypts junk instead of x_C, this will be revealed at the audit date. As the customer certificate contains all necessary information, this fraud can be proven by the customer.

3.3 Proving Correct Usage of the Tracing Keys

The tracing keys are used to encrypt identifying information during withdrawal and payment. To prove the correctness of these encryptions we use the Chaum-Pedersen protocol [CP92], which is an extension of the Schnorr signature scheme [Sch91] and proves the equality of discrete logarithms. As the proof is computed non-interactively, we require a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.

The proof of equality of discrete logarithms $log_{a_1} b_1$ and $log_{a_2} b_2$ is called $PEQDL(a_1, b_1, a_2, b_2) = (c, s)$. This proof can only be given, if the prover knows the discrete logarithm $x = log_{a_1} b_1 = log_{a_2} b_2$. The prover chooses a secret $r \in_R \mathbb{Z}_q$ and computes:

$$c = H(a_1, b_1, a_2, b_2, a_1^r, a_2^r)$$
$$s = r - cx \bmod q$$

Then the verifier has to check, whether c is equal $H(a_1, b_1, a_2, b_2, a_1^s b_1^c, a_2^s b_2^c)$. It is also possible to compute a $PEQDL(m, a_1, b_1, a_2, b_2)$ dependant on a message m, if the hash value is computed as $H(m, a_1, b_1, a_2, b_2, a_1^r, a_2^r)$.

4 Implementation of the Payment System

In this section we describe the implementation of the our payment system and present the protocols for withdrawal, payment, and deposit. Our implementation is based on the off-line payment system of [CMS96, CMS97].

4.1 Withdrawal

For every coin the customer chooses a secret value $\alpha \in_R \mathbb{Z}_q^*$ that is used to compute $(h_p, z_p) = (g_1 g_2^\alpha, y_1 y_2^\alpha)$. This value α has to be escrowed at the bank as $d = y_C^\alpha$ using the customer tracing key y_C. Then the customer receives a blindly issued proof $PEQDL(g, y, h_p, z_p)$ from the bank that the pair (h_p, z_p) has the same discrete logarithm as (g, y). The following withdrawal protocol is also shown in Figure 2.

1. The customer transforms the pair (h_p, z_p) from the payment view to the withdrawal view $(h_w, z_w) = (h_p^{\alpha^{-1}}, z_p^{\alpha^{-1}})$.
2. The customer proves that the pair $((h_w/g_2), g_1)$ has the same discrete logarithm as (y_C, d). Therefore he sends the bank the proof $U = PEQDL((h_w/g_2), g_1, y_C, d)$, h_w, and d. The bank verifies the proof U and only proceeds, if this verification succeeds.
3. The bank and the customer interact in a protocol that blindly issues the proof that (h_w, z_w) from the bank's view respectively (h_p, z_p) from the customer's view has the same discrete logarithm as (g, y).
   - The bank randomly selects a secret $r' \in_R \mathbb{Z}_q$ and computes two commitments $t'_g = g^{r'}$ and $t'_h = h_w^{r'}$, which are sent to the customer.

- The customer chooses blinding factors β and γ and blinds the commitments to $t_g = t'_g g^\beta y^\gamma$ and $t_h = t'^\alpha_h h_p^\beta z_p^\gamma$. Then the customer computes the challenge $c = H(a, g, y, h_p, z_p, t_g, t_h)$, where a random $r \in_R \mathbb{Z}_q$ is used to create the coin number $a = g_2^r$. The challenge is blinded to $c' = c - \gamma \bmod q$ and sent to the bank.
- The bank signs this challenge by computing $s' = r' - c'x \bmod q$ and returns this value to the customer.
- The customer unblinds the response to $s = s' + \beta \bmod q$ so that (c, s) is a valid $PEQDL(a, g, y, h_p, z_p)$.

At the end of this withdrawal the customer stores r, a, α, h_p, z_p, c, and s, while the bank only needs to memorize h_w and d.

Let G be a group of prime order q with the generators g, g_1, g_2

$x \in_R \mathbb{Z}_q$ is the blind signature key of the bank

$y = g^x$, $y_1 = g_1^x$, $y_2 = g_2^x$ are the public keys of the bank

y_C is the public tracing key for this customer

Bank		Customer
		choose $\alpha \in_R \mathbb{Z}_q^*$
		$(h_p, z_p) = (g_1 g_2^\alpha, y_1 y_2^\alpha)$
		$(h_w, z_w) = (h_p^{\alpha^{-1}}, z_p^{\alpha^{-1}})$
		$d = y_C^\alpha$
verify proof U	$\xleftarrow{h_w, d, U}$	$U = PEQDL((h_w/g_2), g_1, y_C, d)$
choose $r' \in_R \mathbb{Z}_q$		
$t'_g = g^{r'}$		
$t'_h = h_w^{r'}$	$\xrightarrow{t'_g, t'_h}$	choose $\beta, \gamma \in_R \mathbb{Z}_q$
		$t_g = t'_g g^\beta y^\gamma$
		$t_h = t'^\alpha_h h_p^\beta z_p^\gamma$
		choose $r \in_R \mathbb{Z}_q$
		$a = g_2^r$
		$c = H(a, g, y, h_p, z_p, t_g, t_h)$
	$\xleftarrow{c'}$	$c' = c - \gamma \bmod q$
$s' = r' - c'x \bmod q$	$\xrightarrow{s'}$	
		$s = s' + \beta \bmod q$
		check $(c, s) \stackrel{?}{=} PEQDL(a, g, y, h_p, z_p)$

Fig. 2. Withdrawal of a coin.

4.2 Payment

At payment the merchant sends the customer a unique value $m = (ID_{merchant}, counter)$ and his public tracing key y_M together with the corresponding merchant certificate of the bank. The customer checks this merchant certificate and proceeds, if it is valid and has not expired.

For every coin the following steps are executed (see also Figure 3):

1. The customer sends a, h_p, z_p, c, and s to the merchant, who verifies that (c, s) is a valid $PEQDL(a, g, y, h_p, z_p)$.
2. To enable owner tracing the customer has to compute the value $d' = y_M^{\alpha^{-1}}$. Then he proves that this value uses the same α as the values h_p from the $PEQDL$ of step 1: He computes $c_p = H(m, d', y_M, g_2, (h_p/g_1), d'^r, a)$ and $s_p = r - c_p\alpha \bmod q$ which are a valid $PEQDL(m, d', y_M, g_2, (h_p/g_1))$. Then d', c_p, and s_p are sent to the merchant.
3. The merchant checks the validity of this $PEQDL(m, d', y_M, g_2, (h_p/g_1))$ and additionally verifies that the same $a = g_2^r$ as committed in the coin's c was used to compute c_p. This forces the customer to use the fixed commitment a so that spending a coin twice will reveal α (see section 5.1).

Finally, the merchant stores a, h_p, z_p, c, s, m, c_p, s_p, and d'.

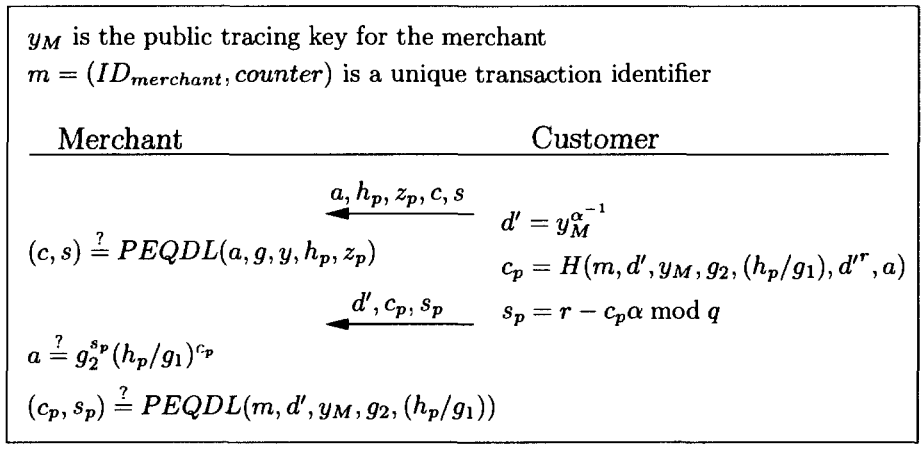


Fig. 3. Off-line payment with a coin.

4.3 Deposit

The merchant forwards all stored values to the bank, who does the same verifications as the merchant during the payment. Note, that the bank has to check everything, even if the merchant is not traced and recovering the withdrawal

view for this coin is not possible. Furthermore, the bank has to ensure that the used y_M belongs to the depositing merchant and that the corresponding merchant certificate is still valid.

5 Tracing and Audit

We describe all tracing methods provided by our payment system and show how coin and owner tracing can be audited.

5.1 Tracing Double-Spenders

At payment the customer has to compute $(c_p, s_p) = PEQDL(m, d', y_M, g_2, (h_p/g_1))$, where he is forced to always use $a = g_2^r$ as the commitment. If he does this twice, the bank receives

$$s_p = r - c_p\alpha \bmod q$$
$$s'_p = r - c'_p\alpha \bmod q$$

from the double-spender. As s_p, s'_p, c_p, and c'_p are known, the bank simply computes $\alpha = (s_p - s'_p)/(c'_p - c_p) \bmod q$. Now the bank looks up, who has withdrawn the coin with $h_w = g_1^{\alpha^{-1}} g_2$, and accuses this person of double-spending. The bank cannot falsely claim that somebody is a double-spender, as it can only compute α after a double-spending.

5.2 Coin and Owner Tracing

Coin tracing is only possible at withdrawal, while owner tracing can only be performed at payment.

- **Coin Tracing:** In the case of coin tracing the customer is traced. The bank knows the private tracing key x_C, which is the discrete logarithm of $y_C = g_2^{x_C}$ to the base g_2. Thus it can "decrypt" $d = y_C^\alpha$ by computing $g_1 d^{x_C^{-1}} = g_1 g_2^\alpha = h_p$. This value reveals the payment view of the coin so that the coin is recognized at deposit.
- **Owner Tracing:** In the case of owner tracing the merchant is traced. The bank knows the private tracing key x_M, which is the discrete logarithm of $y_M = g_1^{x_M}$ to the base g_1. Thus it can "decrypt" $d' = y_M^{\alpha^{-1}}$ by computing $d'^{x_M^{-1}} g_2 = g_1^{\alpha^{-1}} g_2 = h_w$. This value reveals the withdrawal view of the coin so that the identity of the withdrawer can be looked up.

5.3 Self Deanonymization

Self deanonymization is a special form of coin tracing. It can only be applied, if the customer knows x_s, the discrete logarithm of $g_C = g_2^{x_s}$ to the base g_2.

1. The customer requests the value $d = y_C^\alpha$ from the bank, which belongs to one of his withdrawn coins that has to be deanonymized.

2. He removes his secret exponent by computing $d^{x_s^{-1}}$ and returns this value to the bank. To prevent the customer from cheating, he must prove the correctness of his computation by providing the $PEQDL(d^{x_s^{-1}}, d, g_2, g_C)$.
3. If the customer sends the correct value, the bank computes h_p:
   - If the customer has not been traced, then $y_C = g_C^{x_C}$ and the bank computes $g_1(d^{x_s^{-1}})^{x_C^{-1}} = g_1 g_2^{\alpha} = h_p$.
   - If the customer has been traced, then $y_C = g_2^{x_C}$ and the bank computes $g_1 d^{x_C^{-1}} = g_1 g_2^{\alpha} = h_p$.

   If the coin has not been deposited, the bank can blacklist the coin with this value h_p. Otherwise, the bank can look up which merchant deposited the coin.

5.4 Audit

For every audit date the bank publishes a single symmetric key that has been used to encrypt the key generation parameters x_C and x_M in the customer and merchant certificates. Now every customer and merchant with a user certificate of this audit date can detect, whether his certificate was issued for tracing. The published symmetric key enables to decrypt the key generation parameters x_C and x_M so that tracing can be audited as follows:

- Coin tracing is detected by the customer, if y_C is not equal to $g_C^{x_C}$.
- Owner tracing is detected by the customer or merchant, if y_M is not equal to g^{x_M}.

If tracing has been detected, the customer or merchant can request a tracing certificate from the bank. If the bank cannot provide a tracing certificate, the user certificate can be given to a judge. This user certificate has been signed by the bank and contains all necessary information to prove the application of tracing. The judge checks whether tracing indeed has been applied and whether this tracing was legal. The judge denounces any detected illegal tracing.

6 Tracing Capabilities and Privacy

In this section we don't consider payment systems without deanonymization, because they allow misuse of anonymity. For deanonymization two kinds of tracing mechanisms exist:

Self deanonymization: The deanonymization process is under the control of the withdrawer. Thus, tracing can only be used against blackmailers.

Enforced deanonymization: The deanonymization process cannot be influenced by the withdrawer. Thus, tracing can be used for any kind of investigations.

Both mechanisms significantly differ in their anonymity guarantees. Only self deanonymization can provide unconditional anonymity [KV01a] or at least computational anonymity [PS01].

If enforced deanonymization is desired, only weaker anonymity can be guaranteed. Payment systems with a trustee concept (e.g. [CMS96,FTY96,DFTY97, JY96]) have the best tracing capabilities, as tracing is always possible. However, the level of anonymity is uncertain, because tracing is undetectable for the customer and even if tracing has not been applied yet, it still can be applied in the future.

The solution to this problem is the audit concept: Tracing is only possible at either withdrawal or payment and can be detected by the customer after the audit date. Thus, the level of anonymity is only unknown till the audit date. Afterwards an untraced coin remains either unconditional anonymous [KV01b, KV01c] or computational anonymous, which we have presented in this paper. The advantage of this reduction to computational anonymity is the possibility of off-line payments. Furthermore self deanonymization can be applied at any time, as it is not restricted to the withdrawal.

Compared to the directly related off-line payment systems [CMS96,DFTY97] and [PS01] our new solution is clearly superior, as we provide both, a high level of anonymity and at the same time strong deanonymization mechanisms.

7 Conclusion

We have presented the first off-line payment system that provides auditable tracing. In our payment system the bank alone has the power to perform coin and owner tracing. However, the application of tracing is restricted, as the customers and merchants can later audit their payments and detect every deanonymization.

In an alternative implementation of our payment system all the tracing capabilities are transferred to a *tracing authority*, who is responsible for deanonymizations and issues tracing keys and user certificates. The advantage of this implementation is that the tracing capabilities are completely separated from the payment functionality, which allows the bank to focus on the payment system. Anyway, the tracing authority need not be trusted, as the audit concept will reveal any misuse of tracing.

Compared to previous payment systems based on the audit concept, we additionally provide off-line payments and a self deanonymization mechanism that allows the customer to trace his own coins at any time. The self deanonymization is especially useful, if the customer was blackmailed, because the decision about tracing the blackmailed coins is not restricted to the withdrawal.

References

[Cha83] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO '82*, pages 199–203. Plenum, 1983.

[CMS96] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In *Computer Security – ESORICS '96*, volume 1146 of *Lecture Notes in Computer Science*, pages 31–43. Springer-Verlag, 1996.

[CMS97] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. *Journal of Computer Security*, 5(1), 1997.

[CP92] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1992.

[DFTY97] G. Davida, Y. Frankel, Y. Tsiounis, and M. Yung. Anonymity control in e-cash systems. In *Financial Cryptography – FC '97*, volume 1318 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1997.

[FTY96] Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs": Achieving efficient fair off-line e-cash. In *Advances in Cryptology – ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 286–300. Springer-Verlag, 1996.

[JY96] M. Jakobsson and M. Yung. Revokable and versatile electronic money. In *3rd ACM Conference on Computer and Communications Security – CCS '96*, pages 76–87. ACM Press, 1996.

[KV01a] D. Kügler and H. Vogt. Marking: A privacy protecting approach against blackmailing. In *Public Key Cryptography – PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 137–152. Springer-Verlag, 2001.

[KV01b] D. Kügler and H. Vogt. Fair tracing without trustees. In *Financial Cryptography – FC 2001*. Preproceedings, 2001.

[KV01c] D. Kügler and H. Vogt. Auditable tracing with unconditional anonymity. In *Proceedings of the 2nd International Workshop on Information Security Applications (WISA 2001)*, pages 108–120, Seoul, Korea, 2001.

[PS01] B. Pfitzmann and A.-R. Sadeghi. Self-escrowed cash against user blackmailing. In *Financial Cryptography – FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 42–52. Springer-Verlag, 2001.

[Sch91] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch97] B. Schoenmakers. Security aspects of the ecash payment system. In *COSIC '97 Course*, volume 1528 of *Lecture Notes in Computer Science*, pages 338–352. Springer-Verlag, 1997.

[SPC95] M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *Advances in Cryptology – EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 209–219. Springer-Verlag, 1995.

[STS99] T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 555–572. Springer-Verlag, 1999.

[vSN92] B. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581–583, 1992.

Fileteller: Paying and Getting Paid for File Storage

John Ioannidis[1], Sotiris Ioannidis[2], Angelos D. Keromytis[3], and Vassilis Prevelakis[4]

[1] AT&T Labs – Research, *ji@research.att.com*
[2] CIS Department, University of Pennsylvania, *sotiris@dsl.cis.upenn.edu*
[3] CS Department, Columbia University, *angelos@cs.columbia.edu*
[4] MCS Department, Drexel University, *vp@drexel.edu*

Abstract. FILETELLER is a credential-based network file storage system with provisions for paying for file storage and getting paid when others access files. Users get access to arbitrary amounts of storage anywhere in the network, and use a micropayments system to pay for both the initial creation of the file and any subsequent accesses. Wide-scale information sharing requires that a number of issues be addressed; these include distributed access, access control, payment, accounting, and delegation (so that information owners may allow others to access their stored content). In this paper we demonstrate how all these issues are addressed using a micropayment architecture based on a trust-management system. Utilizing the same mechanism for both access control and payment results in an elegant and scalable architecture.

Keywords: Micropayments, trust management, network storage, access control.

1 Introduction

We have set out to create an architecture for a file storage marketplace, which we have called FILETELLER. We envision that there will be Network Storage Providers (NSPs) that make available file storage and charge for it. These storage providers can be traditional ISPs, new businesses created solely for the purpose of providing large amounts of file storage, cooperating organizations, or even individuals with a fast Internet connection and spare capacity on their home systems who want to make such capacity available for a small fee.

The actual price paid for this service can be in real money, in loyalty points, or even in closed-system currency ("play-money" used only as a bookkeeping device among users of the system). We use the trust-management-based micropayments system described in [1] and summarized in Section 1.3 to handle the very small payments that would accompany use of the FILETELLER service. Furthermore, we use a trust-management system for all the access control as well, thereby integrating the payment and the access control mechanisms; access is granted

M. Blaze (Ed.): FC 2002, LNCS 2357, pp. 282–299, 2003.

not only on the basis of who someone is or what credentials they hold, but also on whether they can pay for it.

1.1 Motivation

Probably the most onerous operation for individual computer users is backing up their file systems. Local disks have grown tremendously in size in recent years; it is not uncommon for a home computer to have 50-100GB of disk space. At the same time, the usual backup media (tape and CD-R) have not grown much in capacity, and tape storage in particular is much more expensive relative to the price of disk than it used to be. Commercial endeavors are already offering "network drives" where, for a small monthly fee comparable to what an ISP charges for Internet access, users are entitled to some amount of network-attached storage. We believe that wide availability of such offers, paid for with fixed monthly fees or on a per-use basis at a low per-transaction cost, are going to become increasingly available in coming years. These "network drives" can simply be used as backup storage, or can even be used as the main repository of all user data (backed up by the NSP), while the file system on the user's machine acts as a file cache, much like AFS[2] does.

Off-site backup storage and provisions for disaster recovery, once the purview of large, well-funded organizations, are becoming increasingly important to smaller companies, academic institutions with limited IT budgets, and even individuals. While setting up off-site storage operations for just that purpose may be expensive, groups of similar in nature and geographically separated organizations may want to share their excess capacity. By using a system such as FILETELLER, they can do so in a straightforward way without worrying about one member abusing other members' resources, because of the implicit accounting (even if it is just "play money,") that goes with the system.

Another reason to have network-attached storage is file sharing. Today, if network users wish to make some of their files available to others on the Internet, they can place them on their Web pages and publish the corresponding URL, or in older times "put them up for anonymous FTP." Neither mechanism provides for easy access control or the possibility of financial gain for the owner of the information. Web pages can be password-protected, and FTP directories (even anonymous FTP directories) structured in such a way as to only allow authorized parties to access the information; however, the only users allowed to access the files are those that are already known to the system. This, as is the case with existing network file systems, limits access only between users in the same administration domain and requires considerable involvement of the administrator and the file owner in the access control management. When replication for availability and performance scalability purposes is taken into consideration, the management complexity increases drastically. The WebDAV system [3] solves some of these problems, but its main purpose is group manipulation of shared files, not occasional access.

1.2 Non-goals

It is tempting when designing a new file storage system to attempt to solve all existing problems and issues in network file storage. We have specifically avoided doing that; we are not providing a new name space, a new file access method, a new cryptographic file system, or even a new network file system (in the sense of NFS or CIFS). We are not guaranteeing either integrity or confidentiality of the stored information. We do require the existence of a transport mechanism, such as HTTP, that can carry all the meta-data necessary to accomplish the management functions of FILETELLER, but we have endeavored to rely on existing protocols for the actual data transfer and user/server authentication.

1.3 KeyNote Microchecks

The micropayments system introduced in [1] forms the basis of our approach. The general architecture of this microbilling system is shown in Figure 1. In FILETELLER, the Check Guarantor plays the role of *Provisioning*, the Network User plays the role of *Payer*, and the Network Storage Provider (or another NU acting as an NSP) plays the role of the *Merchant*. *Clearing* is done either by a financial institution (if real money is used) or by a selected user of the system (when loyalty points or "play money" are used).

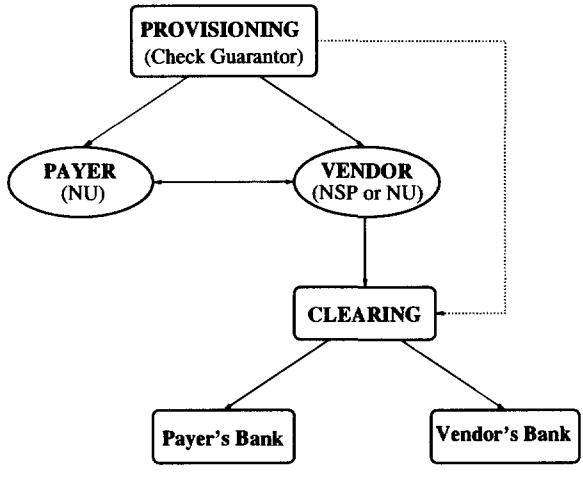


Fig. 1. Microbilling architecture diagram. We have the generic terms for each component, and in parentheses the corresponding players in FILETELLER. The arrows represent communication between the two parties: Provisioning issues credentials to Payers and Merchants; these communicate to complete transactions; Merchants send transaction information to Clearing which verifies the transaction and posts the necessary credits/charges or arranges money transfers. Provisioning and Clearing exchange information on the status of Payer and Merchant accounts.

In this system, Provisioning issues KeyNote[4] credentials to Payers and Merchants. These credentials describe the conditions under which a Payer is allowed to perform a transaction, and the fact that a Merchant is authorized to participate in a particular transaction. When a Payer wants to buy something from a Merchant, the Merchant first encodes the details of the proposed transaction into an *offer* which is transmitted to the Payer.

If the Payer wishes to proceed, she must issue to the Merchant a microcheck for this offer. The microchecks are also encoded as KeyNote credentials that authorize payment for a specific transaction. The Payer creates a KeyNote credential signed with her public key and sends it, along with her Payer credential, to the Merchant. This credential is effectively a check signed by the Payer (the Authorizer) and payable to the Licensee. The conditions under which this check is valid match the offer sent to the Payer by the Merchant. Part of the offer is a nonce, which maps payments to specific transactions, and prevents double-depositing of microchecks by the Merchant.

To determine whether he can expect to be paid (and therefore whether to accept the payment), the Merchant passes the action description (the attributes and values in the offer) and the Payer's key along with the Merchant's policy (that identifies the Provisioning key), the Payer credential (signed by Provisioning) and the microchecks credential (signed by the Payer) to his local KeyNote compliance checker. If the compliance checker authorizes the transaction, the Merchant is guaranteed that Provisioning will allow payment. The correct linkage among the Merchant's policy, the Provisioning key, the Payer key, and the transaction details follow from KeyNote's semantics[4].

If the transaction is approved, the Merchant should give the item to the Payer and store a copy of the microcheck along with the payer credential and associated offer details for later settlement and payment. If the transaction is not approved because the limits in the payer credentials have been exceeded then, depending on their network connectivity, either the Payer or the Merchant can request a transaction-specific credential that can be used to authorize the transaction. Observe that this approach, if implemented transparently and automatically, provides a continuum between online and offline transactions tuned to the risk and operational conditions.

Periodically, the Merchant will 'deposit' the microchecks (and associated transaction details) he has collected to the Clearing and Settlement Center (CSC). The CSC may or may not be run by the same company as the Provisioning, but it must have the proper authorization to transmit billing and payment records to the Provisioning for the customers. The CSC receives payment records from the various Merchants; these records consist of the Offer, and the KeyNote microcheck and credential from the payer sent in response to the offer. In order to verify that a microcheck is good, the CSC goes through the same procedure as the Merchant did when accepting the microcheck. If the KeyNote compliance checker approves, the check is accepted. Using her public key as an index, the payer's account is debited for the amount of the transaction. Similarly, the Merchant's account is credited for the same amount.

This architecture makes it possible to encode risk management rules for micropayments. Previous electronic systems have focused on preventing fraud and failure, rather than on managing it. Unfortunately, the prevention mechanisms can be too expensive for micropayments, making a risk management approach particularly attractive.

2 Architecture

We start by giving a high-level overview of the FILETELLER architecture, and then proceed with a usage example. We close this section with a brief security analysis of our architecture.

2.1 FILETELLER Operation

The main architectural principle behind FILETELLER is that users with properly guaranteed microchecks should be able to 'purchase' storage anywhere they like without any prior arrangements.

There are three participants in this system: *Network Users* (NUs), *Network Storage Providers* (NSPs), and *Check Guarantors* (CGs). All participants are identified by their public keys. An NU holds one or more credentials issued by a CG indicating the NU's credit line with the CG, as shown in Figure 5. As with [1], CG credentials issued to users are relatively short-lived, in order that there be no need for credential revocation lists. We should point out however that any revocation mechanism can be used with our system, specified on a per-credential basis. For the remainder of this discussion, and without loss of generality, we will assume that credential revocation is time-based.

There are four kinds of credentials used in different parts of the system:

1. Check Guarantor credentials, which specify a user's line of credit.
2. Microchecks, which authorize a payment from an NU to an NSP.
3. Microchecks, which authorize a payment from an NU to another NU.

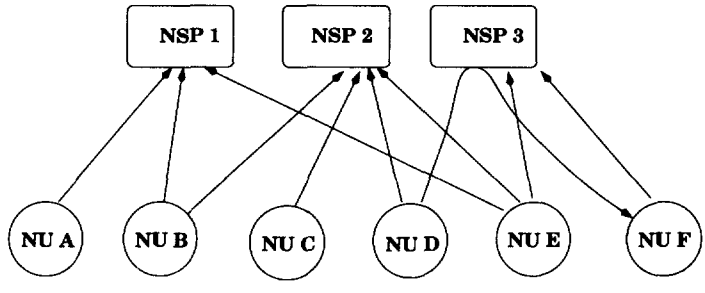


Fig. 2. Flow of payments in our system. Users issue microchecks to the various NSPs as they create and access files, or to other users (through the pay-back scheme).

4. Server credentials, issued by the CGs, that identify complying NSPs the NUs can use.
5. File access credentials, issued by NSPs when the file is created, authorizing subsequent access to that file.
6. File access credentials, issued by NUs that own a file, authorizing access to that file.

NUs interact with network storage providers using some file access protocol. The details of the protocol are not important for our architecture; a distributed authoring system such as WebDAV, a file transfer protocol such as HTTP or FTP, or any other data transfer protocol can be used. We assume that the granularity of access is the file; that is, users create, read, delete, append to, or replace whole files. We work with whole files rather than individual blocks for two reasons: to amortize the cost of a check verification over the transfer of an entire file, and to avoid choosing some arbitrary block size and defining block-level operations, which would probably tie FILETELLER to a particular file system philosophy rather than make it a general file-storage service.[1]

NUs may also interact with each other in exchanging files; the role of NSP is an assumed one, and has no physical significance in our architecture. Thus, our architecture can operate in a peer-to-peer environment. Naturally, one would expect that commercial NSPs have better connectivity and storage capacity, and offer better availability guarantees than average users. The data-flows in our system are shown in Figure 4.

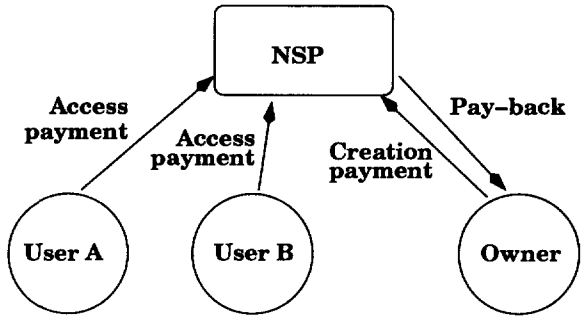


Fig. 3. File owners submit a payment to the NSP at file creation time, and at each access thereafter. Other users also submit payments to the NSP in order to access the file (assuming they have been authorized by the owner). The owner may receive a pay-back from the users through the NSP, if he defines the appropriate file disposition.

After a connection has been established, possibly secured using IPsec or TLS, the client describes the transaction to the server. In particular, the client

[1] That said, a layer can always be added on top of FILETELLER that sends and receives what to FILETELLER look like small, fixed-size files named with sequential numbers, which the layer treats as blocks. We advise against this.

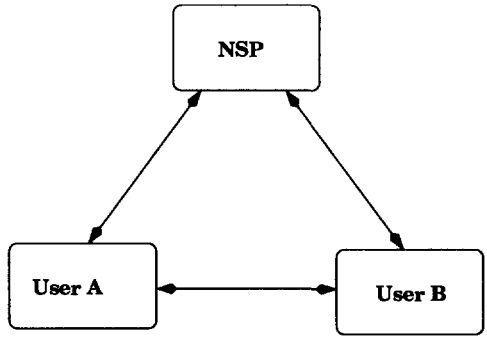


Fig. 4. File data can be stored by "official" NSPs, or by users with spare capacity (*e.g.*, peer-to-peer scenario). Any user in our system can assume the role of an NSP, allowing for a very dynamic storage market, with a low barrier-to-entry.

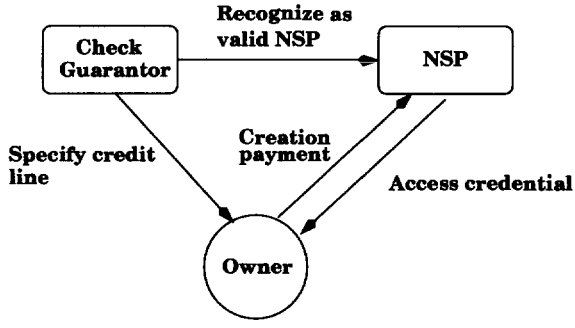


Fig. 5. NSPs issue a credential to each Check Guarantor (CG) authorizing them to act as introducers of users, by issuing them a KeyNote credential. A file owner needs to convince a CG to provide them with a credit line, expressed as a KeyNote credential. The file owner needs to provide these two credentials to the NSP, along with a microcheck conveying payment to the NSP. In response, the NSP returns to the file owner a KeyNote access credential, granting her full privileges in accessing the file.

needs to tell the NSP the name of the file to be accessed, the operation type (create, replace, read, append, delete), the size of the file (for replace, append, or create), the file disposition (ordinary or "pay-back"), the CG credential(s), the ID of the transaction (a random number that the client uses to match requests with responses), and who the owner of the file will be (if someone other than the creator). When the NSP receives this description, it makes sure that the CG is one that it knows about, and sends back an *offer*, consisting of the following fields:

- Transaction ID (echoes what was sent).
- Nonce (for billing purposes).
- Cost, in a real or "play" currency.

The client then writes a microcheck, and sends it along with the credentials necessary to approve the operation: the CG credential(s) and, if this is an access to an existing file, the file ownership chain of credentials, followed by a copy of the offer, the operation, the file disposition, and the attributes, as shown in Figure 6. If the operation was a 'create,' 'replace,' or 'append,' the contents of the file are sent. If the operation was a 'read,' or a 'delete,' nothing else is sent.

When a file is created, the server responds with a file access credential, authorizing the creator of the file to full access (provided the payment has been made). If a file is replaced or appended to, the server responds with a signed 'receipt' for the transaction. Otherwise, the file is sent to the client.

If the client request was for a file with a "pay-back" disposition, two microchecks must be issued; one to the server, and one to the owner. The server will verify that the owner will get paid before releasing the file. The server can either deposit these on the user's behalf, or give them to the owner (in the form of microchecks) for processing. The flow of payments in our architecture is shown in Figures 2 and 3.

There is little difference between processing microchecks drawn in a real currency or in "play money." In the former case, the microchecks must eventually be processed by a financial institution (a bank), as in [1]. In the case of play money, some agent (one of the CGs, or a mutually-agreed-upon party) collects all the microchecks and processes them according to the internal accounting procedures of the group. In the case of real money, market forces and competition will set the fair price for the service.

File attributes are used in file access delegation credentials to allow subsets of a user's files to be shared. These attributes are meta-data associated with the file by the owner, and can be used to implement easy file grouping, associate security labels with files, or for any other similar scheme.

2.2 Example Usage Scenario

Having seen the overall system architecture, let us look at a particular example. *Alice* is a user who wants to store some of her files in *Nick's* Network Storage Provider servers. Every morning Alice contacts her banker and obtains a fresh *Check Guarantor* credential, which allows her to write KeyNote microchecks. The CG credential (most of the hex digits from the keys have been removed for brevity) allows Alice to write checks for up to 5 US Dollars, and she can do so until March 24th, 2002.

```
Keynote-Version: 2
Local-Constants: ALICE_KEY = "rsa-base64:MCgCIQGBOf8lSVZfHDwdck\
      ESR/Dh+ONPMrYvdOQlU9QdKbKbRQIDAQAB"
      CG_KEY = "rsa-base64:MIGJAo..."
Authorizer: CG_KEY
Licensees: ALICE_KEY
Conditions: app_domain == "FileTellerPay" && currency == "USD"
      && &amount < 5.01 && date < "20020324" -> "true";
Signature: "sig-rsa-sha1-base64:QU6SZtG9R3IXXAU9vRDBgu..."
```

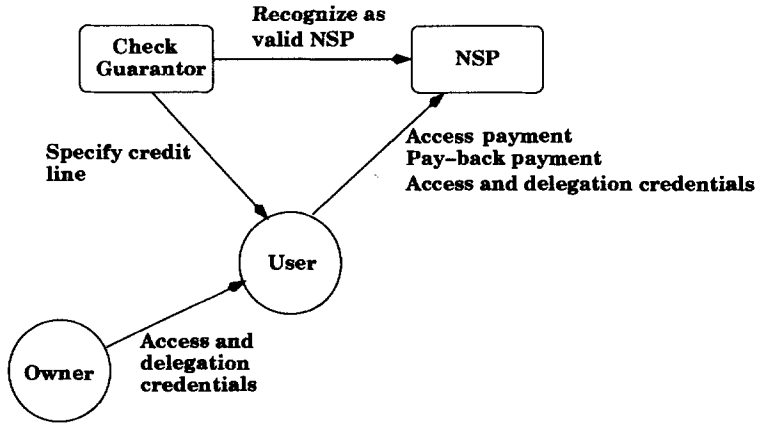


Fig. 6. A user wishing to access another user's file needs to have their own line of credit with a Check Guarantor (CG), as well as a credential from the file owner granting them access to that file. When accessing the file, the user needs to provide the credit-backing credential from the CG, a microcheck to the NSP, and the access credential(s) to the file. If the owner has set a "pay-back" disposition for the file, an additional microcheck to the owner may also be needed to gain access.

Alice now wants to store last night's backup, which took up 250 Mbytes. At the rate of $0.001/Mbyte, she knows that she has to pay 25 cents to store it. She writes a check for this amount:

```
Keynote-Version: 2
Local-Constants: ALICE_KEY = "rsa-base64:Mcg..."
      NICK_KEY = "rsa-base64:MCgQGB..."
Authorizer: ALICE_KEY
Licensees: NICK_KEY
Conditions: app_domain == "FileTellerPay" &&
      currency == "USD" && amount == "0.25" &&
      nonce == "eb2c3dfc860dde9a" && date == "20020123" -> "true";
Signature: "sig-rsa-sha1-base64:Qpf..."
```

The `nonce` is a random number that must be different for each check, guaranteeing that there will be no double-depositing of checks. She also constructs a request to Nick telling him that she is about to send over a 250 Mbyte file, gives the file name, and sends it along with the microcheck and the Check Guarantor credential. Note that Alice uses the same key to identify Nick in his function as a recipient of money, and in his function as the storage server. Actually, a simple string such as "NICK'S DISK FARMS" would have sufficed for the purposes of writing the microcheck. However, it makes bookkeeping easier to know each principal by their public key.

Nick receives these credentials, validates the microcheck to make sure that he will get paid, and waits for the file to arrive. If the check is not good, Nick

will say so, and refuse to accept the file. Of course, Nick may keep the check and refuse the file anyway, which just means that Alice will probably accept the loss of 25 cents, and never do business with Nick again. If Nick is honest, he will respond by sending back a *file access credential:*

```
Keynote-Version: 2
Local-Constants: NICK_KEY = "rsa-base64:MCgQGB..."
      ALICE_KEY = "rsa-base64:MIGJAo..."
Authorizer: NICK_KEY
Licensees: ALICE_KEY
Conditions: app_domain == "FileTellerAccess"
      && filename == "/home/alice/backups/fulbak-20020122"
      && ( (access == "R") || (access == "RP") || access == "A" ||
            (access == "D" ) ) -> "true";
Signature: "sig-rsa-sha1-base64:QU634865a88..."
```

Alice can use this credential to access the file, by sending it along with proper payment. The credential grants her Read, Replace, Append, and Delete access; that is, all the operations she may want to perform. If Alice wants to read the file, she will have to send both a microcheck (along with the CG credential) and the file access credential. Nick will verify that he will get paid, and then will evaluate the access credential on the file access credential he got from Alice, in conjunction with his own policy:

```
Keynote-Version: 2
Local-Constants: NICK_KEY = "rsa-base64:MCgQGB..."
Authorizer: POLICY
Licensees: NICK_KEY
Conditions: app_domain == "FileTellerAccess" -> "true";
```

This policy says that anything that Nick's key authorizes is allowed. Since the file was paid for, and a path can be found from POLICY to ALICE_KEY (the requestor of the operation), the operation is allowed.

As a matter of business practice, Bob may require periodic payments from Alice in order to keep her files around. Alice must know that and send microchecks at the appropriate intervals.

Alice may also grant access to Bob to just read one of her files (*e.g.*, if the file were a picture and not a backup). She can *delegate* to Bob by issuing the following credential:

```
Keynote-Version: 2
Local-Constants: ALICE_KEY = "rsa-base64:MIGJAo..."
      BOB_KEY = "rsa-base64:MCgQGXW..."
Authorizer: ALICE_KEY
Licensees: BOB_KEY
Conditions: app_domain == "FileTellerAccess"
      && filename == "/home/alice/pics/DSCN1033.JPG"
      && access == "R" -> "true";
Signature: "sig-rsa-sha1-base64:QU63486532a..."
```

Bob will have to first pay for the access, and send both Alice's original file access credential (that Nick issued) and the one that Alice issued him. In the standard KeyNote way, since a path will be found from Nick's POLICY to BOB_KEY (the requestor), the file will be given to Bob.

In another mode of operation, Alice may want to get paid every time Bob (or anyone else) accesses her files. She indicates this by giving Bob a slightly different credential, and telling him that he must pay 50 cents to access the file:

```
Keynote-Version: 2
Local-Constants: ALICE_KEY = "rsa-base64:MIGJAo..."
      BOB_KEY = "rsa-base64:MCgQGXW..."
Authorizer: ALICE_KEY
Licensees: BOB_KEY
Conditions: app_domain == "FileTellerAccess"
      && filename == "/home/alice/pics/DSCN1033.JPG"
      && access == "R" && value >= "0.50" && currency == "USD" -> "true";
Signature: "sig-rsa-sha1-base64:QU63486532a..."
```

Bob must then issue his request to Nick by also including a microcheck payable to Alice. When Nick gets that microcheck, he includes in the KeyNote *action environment* the price paid, which will enable that last credential to evaluate to `true`. Note that by simply setting the value paid to 0 when he does not receive payment for Alice, Nick can use the exact same mechanism for both kinds of access, and at the same time Alice can have some people get the file for free (by issuing a credential without the payment requirement) or for pay (by issuing a credential with the payment requirement). In either case, Nick will include the amount paid; if Bob does not send payment for Alice but has the latter kind of credential, he will not get access because the `value >= "0.50"` condition will not be satisfied. As part of the service to Alice, Nick collects all the microchecks issued for her and gives them to her at a later time.

Alice does not have to issue individual credentials to each potential customer. She can simply publish the following credential on her Web page:

```
Keynote-Version: 2
Local-Constants: ALICE_KEY = "rsa-base64:MIGJAo..."
Authorizer: ALICE_KEY
Conditions: app_domain == "FileTellerAccess"
      && filename == "/home/alice/pics/DSCN1033.JPG"
      && access == "R" && value >= "0.50" && currency == "USD" -> "true";
Signature: "sig-rsa-sha1-base64:QU63486532a..."
```

In this case, since no Licensees are listed, anyone can use this credential to access the file, provided of course that they pay for it.

Alternately, Alice may wish to pre-pay some fixed amount to a CG on a monthly basis and get some number of transactions for free. In that case, the CG can issue a credential of this form:

```
Keynote-Version: 2
Local-Constants: ALICE_KEY = "rsa-base64:MCgCIQGBOf8lSVZfHDwdck\
      ESR/Dh+ONPMrYvdOQlU9QdKbKbRQIDAQAB"
      CG_KEY = "rsa-base64:MIGJAo..."
Authorizer: CG_KEY
Licensees: ALICE_KEY
Conditions: app_domain == "FileTellerPay" && currency == "USD"
      && (&amount < 5.01 || amount == "prepay") && date < "20020424" -> "true";
Signature: "sig-rsa-sha1-base64:QU6SZtG9R3IXXAU9vRDBgu..."
```

In this case, Alice will tell the NSP that she is using a pre-paid plan when accessing her files, and issue a microcheck authorizing charging of the pre-paid plan. While that microcheck does not directly convey any value transfer, it acts as a proof of the transaction that the NSP can later use to collect payment from the CG, or that a pre-set limit was exceeded.

Finally, Alice may wish to be charged for all accesses to her files, instead of having the users be charged; this may be the case for a company that uses exclusively off-site storage for its operations. There are two ways Alice can achieve this:

- By using a pre-paid plan, as described above.
- By becoming a Check Guarantor and establishing agreements with one or more NSPs. In this case, Alice issues her own CG credentials to her users. If desired, she can require that any files created using these credentials be assigned to her: in that case, the access credential returned by the NSP will specify Alice as the file owner. She can then issue an access credential to the user who created the file, granting them the relevant access. If the user leaves the company, Alice remains the owner of the file, and can authorize someone else to access it.

2.3 Security Analysis

Similar to [1], our system has four types of communication: provisioning, reconciliation, transaction, and delegation. We shall not worry about any value

transfers to banks, as there already exist well-established systems for handling those. All communications between CGs, NSPs, and NUs can be protected with existing protocols such as IPsec or TLS.

The security of the stored data itself is not within the purview of our system, nor is it a responsibility of the NSP; if the user does not trust the NSP not to look at their data, they need to take appropriate measures, such as encrypting their files before storing them at the NSP. Existing mechanisms, such as CFS, SFS, or manually-encrypted files can be used with our architecture to address such concerns.

User provisioning must also be protected; although the credentials themselves are not confidential and are integrity-protected (by virtue of being digitally signed), the mechanism by which the user pays the CG (or somehow establishes trust) must be secured. Such mechanisms however already exist, *e.g.*, online payments using a credit card, and can be used for provisioning.

The user needs to authenticate with the NSP before any file operation can take place. The details of the authentication protocol used are not important, and existing security protocols can be used for that purpose. The requirements for the authentication protocol are that it provides strong authentication and, optionally, lets the user piggy-back credential delivery to the NSP; conveniently, most security protocols in use today (IPsec, TLS) meet these requirements.

At a higher level of abstraction, the NU has to worry about the NSP providing service; this is something that can be easily verified by the NU. Furthermore, the NU can use multiple NSPs to achieve better reliability through replication. Protecting against over-charging NSPs or CGs is also straightforward: the details of each transaction can be verified at any point in time, by verifying the credentials and the offer. Since only the NU can create microchecks, a dispute claim can be resolved by running the transaction again. The NU is safe even from a collusion between CGs and NSPs.

Since the value of the individual transactions is small, losses can be tolerated on either side; in particular, if the NU requests access to a file for which she does not have permission, the NSP can keep the payment[2]. Charging for access to non-existant files depends on the NSP's policy.

The NSP must also faithfully enforce the access control policy of the file owner. This is of particular interest in the case of the "pay-back" scheme. There is no way of guaranteeing this to the NU, except by introducing an encryption layer; in that case, however, the owner will have to be involved in every transaction involving the file. This also raises digital rights management (DRM) issues, which are outside the scope of our architecture. We believe that such issues are better addressed via real-world means (*e.g.*, contracts, legal action, *etc.*).

The NSP must make sure they are paid for the services offered. Since it has a copy of all transactions (the CG credential, the microcheck, and the offer), it can prove to the CG, or any other party, that a transaction was in fact performed.

[2] To our knowledge, this is the first system where illegal file access results in a "fine" for the misbehaving user.

The CG also needs to be paid for the services offered. Since the CG does the clearing of the microchecks, the NSP has to provide the transaction logs to the CG. Using those, the CG can verify that a transaction was done, and at what value. A collusion between the NSP and an NU is somewhat self-contradicting: the NU's goal is to minimize cost, while the NSP's is to maximize revenue, each at the expense of the other. The function of the CG is to verify each transaction (perhaps sampling, for very large numbers of transactions), debit the NSP and credit the NU (presumably keeping some commission or small fee in the process): if the NSP does not give any credentials to the CG, then no work was done as far as the CG is concerned (and no payments are made, which benefits the NU); claiming more transactions than really happened is not in the best interest of the NU (so no collaboration could be expected in the direction), and the NSP cannot "fabricate" transactions.

Since value is not stored in either the NSP or the NU, only a reliable log of the transactions is needed at the NSP (and, optionally, at the NU).

3 Implementation

We have constructed one demonstration system and are working on a second; each implements credential-based access to on-line information using a different protocol. The existing system uses the ftp protocol, while the other will use a Java applet to implement the client, thus providing Web-based access to FILETELLER.

Our prototype implements the functions we described earlier in the Section 2, namely the backup and retrieval of files stored at the remote server, the creation of new files, and the delegation of access rights to other users.

When a file is first introduced to the remote server, a credential allowing its future retrieval is stored on the local machine. To avoid these credential files from cluttering up the user's directories, we place them in a special subdirectory (called FT) in the directory where our file is stored.

Other information such as the user's public and private keys and the keys of remote servers and check guarantors are placed in the *.fileteller* subdirectory in the user's home directory.

To retrieve a file, the user must specify the name of the file (on the local machine) and the directory it was stored in (so that the file "FT/filename,cred" may be located). Other information, such as the name of the server and the name of the file on the server are stored in the credential file).

The most complex operation is that of delegation. Using the DELEGATE command, the user can grant access rights to another user. The delegate command must include the file where the public key of the other user is stored, the permissions (read, write, append, *etc.*) that the other user will be granted, the file name and its directory (on the local machine) and, optionally, the price of the file. This price is the pay-back fee that the local user will receive each time the file is accessed by the remote user, and is in addition to the access fee charged by the storage provider.

This prototype is highly portable and runs on all the platforms that support the KeyNote library. Nevertheless, in order to make the FILETELLER system truly platform-independent, we are working on a Java implementation that will run on all popular Web browsers. The capabilities of the FILETELLER-Java prototype will be similar to those of the existing prototype and a lot of code will be simply ported to Java.

3.1 Scalability

A key characteristic of the system is that it scales very well. To understand why this is true we will present a small example:

Imagine a building with n doors. People wishing to enter the building show up at one of the doors. Any door will do.

The traditional approach specifies that each door has a guard that examines the persons's identification (authentication) and checks the list of people that are allowed to enter the building (ACL, Unix file permissions, etc.). If you are on the list you are in, otherwise, the request is denied.

In the case of FILETELLER, the guards are replaced with locks on the doors. Each person has a key and that key grants access to the building. Assume for a moment that all visitors have the same key. Then any visitor can enter through any door.

Scalability is demonstrated thus: To scale for many visitors, you have to increase the number of doors. In the case of the traditional approach you have the problem of distributing the lists to all the guards. Each guard must have the complete list, so the list distribution overhead, plus the size of the list itself restricts scalability.

In our system, you just add locked doors. Complexity does not depend on the number of doors. Also since each visitor is supplying the key, the complexity of the locks on each door is independent of the total number of visitors or the number of other doors. As the complexity of the mechanism increases the throughput per door may go down, but this can be fixed by adding more doors.

In the FILETELLER system each user has a different key, but the key is authorised by a chain of credentials that lead to the administrator (or owner) of the site. Thus the entrypoints to the system do not have to maintain any state about the users of the site; user supply necessary credentials to establish a path to the authorized key.

4 Related Work

A number of protocols have been deployed for users to share files across the network, the most commonly used being FTP and HTTP [5,6]. Anonymous FTP, where there is no need for authentication, offers great flexibility since any user can download or upload files to FTP servers. Similarly in the Web architecture, access is either anonymous or subject to some sort of ad-hoc authentication mechanism. This configuration is useful only in the case where file content is

non-critical. Where authentication is required, flexibility is greatly reduced since the only users allowed to access the server, in that case, are users who are already known to the system. Furthermore, these schemes lack the ability to charge for file usage which is what FILETELLER is designed to do.

Network file systems (*e.g.,* NFS and AFS [7,2]) are also popular and widespread mechanisms for sharing files within the same administration domain. Crossing administrative boundaries creates numerous administrative problems (*e.g.,* merging distinct Kerberos realms or NIS domains).

Encrypting file systems like CFS [8] place great emphasis on maintaining the privacy of the user information by encrypting the file names and their contents. The limitation of such systems is that sharing is particularly difficult to implement; the file owner must somehow communicate the secret encryption key for the file to all the users that wish to access it. Even then, traditional access controls must still be used to enforce access restrictions (*e.g.,* read-only, append-only, immutable file, *etc.*).

WebFS is part of the larger WebOS [9] project at UC Berkeley. It implements a network file system on top of the HTTP protocol. WebFS relies on user-level HTTP servers used to transfer data, along with a kernel module that implements the file system. Access control lists (ACLs) are associated with each file that enumerate users who have read, write, or execute permission on individual files. Users are uniquely identified by their public keys. We have taken a more general and scalable approach in that there is no need for ACLs since each credential is sufficient to identify both the users and their privileges.

The secure file system (SFS [10]) introduces the notion of *self-certifying pathnames*: file names that effectively contain the appropriate remote server's public key. In this way, SFS needs no separate key management machinery to communicate securely with file servers. However, since SFS relies on a trusted third party to mutually authenticate server and client, collaboration is possible only if the client and the server have a common root for their Certification Authorities.

FILETELLER differs from the file system solutions proposed above in two very significant ways. It uses credentials to identify both the files stored in the file system and the users that are permitted to access them, as well as the circumstances under which such access is allowed. Also, users can delegate access rights simply by issuing new credentials, providing a natural and very scalable way of sharing information. Furthermore, it departs from all previous approaches by incorporating a micropayment scheme that allows users to buy file system space on remote servers and subsequently sell access to their files to anyone on the network.

The use of trust management to support market-like mechanisms was first suggested in [11], in the specific case of an active network architecture. KeyNote credentials were used for payment authorization as well as for encapsulating fine-grained policy rules for controlling system and network resources. Additionally, entities called *service brokers* were introduced for facilitating the exchange of payment credentials and access credentials between *producers* (*e.g.* the network

node) and *consumers* (*e.g.* the untrusted modules). FILETELLER addresses the issue of large scale file sharing, with the additional provision for payments.

Finally, WebDAV [3] is a system that addresses the issue of collaborative authoring. WebDAV defines the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable. It leverages the success of HTTP in being a standard access layer for a wide range of storage repositories – standard HTTP is used for read access, while DAV provides write access. FILETELLER is not limited to a specific access layer and is able to work on any file transport substrate, providing support for flexible file charging and access control at the same time.

5 Summary and Concluding Remarks

We have presented FILETELLER, a system for flexible global file sharing, made possible by using a credential-based network file storage system with provisions for paying for file storage and getting paid when others access files. Both the file access control and payment are implemented using the KeyNote trust-management system,

Using a trust-management system as the access-control mechanism relieves the servers from having to keep access control lists or other authorization data, since all the necessary information is kept in a distributed fashion by each individual user in the form of KeyNote credentials. The use of a micropayments system allows each request to be paid-for as it occurs, and the only bookkeeping that has to happen is the clearance of the microchecks; this further reduces overheads on the server.

Our plans for future research include the integration of FILETELLER with the WebDAV protocols, with the ultimate objective to create an academic research portal site acting as a vehicle for collaborative authoring of research papers and repository of published work.

Acknowledgements. This work was supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795, and DARPA under Contracts F39502-99-1-0512-MOD P0001 and F30602-01-2-0537. We would also like to thank Jonathan Smith for his for valuable comments and suggestions throughout the course of this work.

References

1. Blaze, M., Ioannidis, J., Keromytis, A.D.: Offline Micropayments without Trusted Hardware. In: Proceedings of the Fifth International Conference on Financial Cryptography. (2001)
2. Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J.: Scale and performance in a distributed file system. ACM Transactions on Computer Systems **6** (1988) 51–81

3. Whitehead, E.: World Wide Web Distributed Authoring and Versioning (WebDAV): An Introduction. ACM StandardView **5** (1997) 3–8
4. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The KeyNote Trust Management System Version 2. Internet RFC 2704 (1999)
5. Postel, J., Reynolds, J.: File Transfer Protocol. Internet RFC 959 (1985)
6. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. Internet RFC 2069 (1997)
7. Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B.: Design and implementation of the sun network file system. In: Proceedings of the 1985 Summer Usenix Conference, Portland, OR (1985)
8. Blaze, M.: A Cryptographic File System for Unix. In: Proc. of the 1st ACM Conference on Computer and Communications Security. (1993)
9. Vahdat, A.: Operating System Services for Wide-Area Applications. PhD thesis, University of California, Berkeley (1998)
10. Mazieres, D., Kaminsky, M., Kaashoek, M.F., Witchel, E.: Separating key management from file system security. In: Symposium on Operating Systems Principles. (1999) 124–139
11. Anagnostakis, K.G., Hicks, M.W., Ioannidis, S., Keromytis, A.D., Smith, J.M.: Scalable Resource Control in Active Networks. In: Proceedings of the Second International Working Conference on Active Networks (IWAN). (2000) 343–357

Author Index

Lecture Notes in Computer Science

For information about Vols. 1–2501

please contact your bookseller or Springer-Verlag

Vol. 2357: M. Blaze (Ed.), Financial Cryptography. Proceedings, 2002. VIII, 301 pages. 2003.

Vol. 2502: D. Gollmann, G. Karjoth, M. Waidner (Eds.), Computer Security – ESORICS 2002. Proceedings, 2002. X, 281 pages. 2002.

Vol. 2503: S. Spaccapietra, S.T. March, Y. Kambayashi (Eds.), Conceptual Modeling – ER 2002. Proceedings, 2002. XX, 480 pages. 2002.

Vol. 2504: M.T. Escrig, F. Toledo, E. Golobardes (Eds.), Topics in Artificial Intelligence. Proceedings, 2002. XI, 432 pages. 2002. (Subseries LNAI).

Vol. 2506: M. Feridun, P. Kropf, G. Babin (Eds.), Management Technologies for E-Commerce and E-Business Applications. Proceedings, 2002. IX, 209 pages. 2002.

Vol. 2507: G. Bittencourt, G.L. Ramalho (Eds.), Advances in Artificial Intelligence. Proceedings, 2002. XIII, 418 pages. 2002. (Subseries LNAI).

Vol. 2508: D. Malkhi (Ed.), Distributed Computing. Proceedings, 2002. X, 371 pages. 2002.

Vol. 2509: C.S. Calude, M.J. Dinneen, F. Peper (Eds.), Unconventional Models in Computation. Proceedings, 2002. VIII, 331 pages. 2002.

Vol. 2510: H. Shafazand, A Min Tjoa (Eds.), EurAsia-ICT 2002: Information and Communication Technology. Proceedings, 2002. XXIII, 1020 pages. 2002.

Vol. 2511: B. Stiller, M. Smirnow, M. Karsten, P. Reichl (Eds.), From QoS Provisioning to QoS Charging. Proceedings, 2002. XIV, 348 pages. 2002.

Vol. 2512: C. Bussler, R. Hull, S. McIlraith, M.E. Orlowska, B. Pernici, J. Yang (Eds.), Web Services, E-Business, and the Semantic Web. Proceedings, 2002. XI, 277 pages. 2002.

Vol. 2513: R. Deng, S. Qing, F. Bao, J. Zhou (Eds.), Information and Communications Security. Proceedings, 2002. XII, 496 pages. 2002.

Vol. 2514: M. Baaz, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. Proceedings, 2002. XIII, 465 pages. 2002. (Subseries LNAI).

Vol. 2515: F. Boavida, E. Monteiro, J. Orvalho (Eds.), Protocols and Systems for Interactive Distributed Multimedia. Proceedings, 2002. XIV, 372 pages. 2002.

Vol. 2516: A. Wespi, G. Vigna, L. Deri (Eds.), Recent Advances in Intrusion Detection. Proceedings, 2002. X, 327 pages. 2002.

Vol. 2517: M.D. Aagaard, J.W. O'Leary (Eds.), Formal Methods in Computer-Aided Design. Proceedings, 2002. XI, 399 pages. 2002.

Vol. 2518: P. Bose, P. Morin (Eds.), Algorithms and Computation. Proceedings, 2002. XIII, 656 pages. 2002.

Vol. 2519: R. Meersman, Z. Tari, et al. (Eds.), On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE. Proceedings, 2002. XXIII, 1367 pages. 2002.

Vol. 2521: A. Karmouch, T. Magedanz, J. Delgado (Eds.), Mobile Agents for Telecommunication Applications. Proceedings, 2002. XII, 317 pages. 2002.

Vol. 2522: T. Andreasen, A. Motro, H. Christiansen, H. Legind Larsen (Eds.), Flexible Query Answering. Proceedings, 2002. XI, 386 pages. 2002. (Subseries LNAI).

Vol. 2523: B.S. Kaliski Jr., Ç.K. Koç, C. Paar (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2002. Proceedings, 2002. XIV, 612 pages. 2002.

Vol. 2525: H.H. Bülthoff, S.-Whan Lee, T.A. Poggio, C. Wallraven (Eds.), Biologically Motivated Computer Vision. Proceedings, 2002. XIV, 662 pages. 2002.

Vol. 2526: A. Colosimo, A. Giuliani, P. Sirabella (Eds.), Medical Data Analysis. Proceedings, 2002. IX, 222 pages. 2002.

Vol. 2527: F.J. Garijo, J.C. Riquelme, M. Toro (Eds.), Advances in Artificial Intelligence – IBERAMIA 2002. Proceedings, 2002. XVIII, 955 pages. 2002. (Subseries LNAI).

Vol. 2528: M.T. Goodrich, S.G. Kobourov (Eds.), Graph Drawing. Proceedings, 2002. XIII, 384 pages. 2002.

Vol. 2529: D.A. Peled, M.Y. Vardi (Eds.), Formal Techniques for Networked and Distributed Sytems – FORTE 2002. Proceedings, 2002. XI, 371 pages. 2002.

Vol. 2531: J. Padget, O. Shehory, D. Parkes, N. Sadeh, W.E. Walsh (Eds.), Agent-Mediated Electronic Commerce IV. Proceedings, 2002. XVII, 341 pages. 2002. (Subseries LNAI).

Vol. 2532: Y.-C. Chen, L.-W. Chang, C.-T. Hsu (Eds.), Advances in Multimedia Information Processing – PCM 2002. Proceedings, 2002. XXI, 1255 pages. 2002.

Vol. 2533: N. Cesa-Bianchi, M. Numao, R. Reischuk (Eds.), Algorithmic Learning Theory. Proceedings, 2002. XI, 415 pages. 2002. (Subseries LNAI).

Vol. 2534: S. Lange, K. Satoh, C.H. Smith (Ed.), Discovery Science. Proceedings, 2002. XIII, 464 pages. 2002.

Vol. 2535: N. Suri (Ed.), Mobile Agents. Proceedings, 2002. X, 203 pages. 2002.

Vol. 2536: M. Parashar (Ed.), Grid Computing – GRID 2002. Proceedings, 2002. XI, 318 pages. 2002.

Vol. 2537: D.G. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. Proceedings, 2002. VII, 237 pages. 2002.

Vol. 2538: B. König-Ries, K. Makki, S.A.M. Makki, N. Pissinou, P. Scheuermann (Eds.), Developing an Infrastructure for Mobile and Wireless Systems. Proceedings 2001. X, 183 pages. 2002.

Vol. 2539: K. Börner, C. Chen (Eds.), Visual Interfaces to Digital Libraries. X, 233 pages. 2002.

Vol. 2540: W.I. Grosky, F. Plášil (Eds.), SOFSEM 2002: Theory and Practice of Informatics. Proceedings, 2002. X, 289 pages. 2002.

Vol. 2541: T. Barkowsky, Mental Representation and Processing of Geographic Knowledge. X, 174 pages. 2002. (Subseries LNAI).

Vol. 2542: I. Dimov, I. Lirkov, S. Margenov, Z. Zlatev (Eds.), Numerical Methods and Applications. Proceedings, 2002. X, 174 pages. 2003.

Vol. 2544: S. Bhalla (Ed.), Databases in Networked Information Systems. Proceedings 2002. X, 285 pages. 2002.

Vol. 2545: P. Forbrig, Q, Limbourg, B. Urban, J. Vanderdonckt (Eds.), Interactive Systems. Proceedings 2002. XII, 574 pages. 2002.

Vol. 2546: J. Sterbenz, O. Takada, C. Tschudin, B. Plattner (Eds.), Active Networks. Proceedings, 2002. XIV, 267 pages. 2002.

Vol. 2547: R. Fleischer, B. Moret, E. Meineche Schmidt (Eds.), Experimental Algorithmics. XVII, 279 pages. 2002.

Vol. 2548: J. Hernández, Ana Moreira (Eds.), Object-Oriented Technology. Proceedings, 2002. VIII, 223 pages. 2002.

Vol. 2549: J. Cortadella, A. Yakovlev, G. Rozenberg (Eds.), Concurrency and Hardware Design. XI, 345 pages. 2002.

Vol. 2550: A. Jean-Marie (Ed.), Advances in Computing Science – ASIAN 2002. Proceedings, 2002. X, 233 pages. 2002.

Vol. 2551: A. Menezes, P. Sarkar (Eds.), Progress in Cryptology – INDOCRYPT 2002. Proceedings, 2002. XI, 437 pages. 2002.

Vol. 2552: S. Sahni, V.K. Prasanna, U. Shukla (Eds.), High Performance Computing – HiPC 2002. Proceedings, 2002. XXI, 735 pages. 2002.

Vol. 2553: B. Andersson, M. Bergholtz, P. Johannesson (Eds.), Natural Language Processing and Information Systems. Proceedings, 2002. X, 241 pages. 2002.

Vol. 2554: M. Beetz, Plan-Based Control of Robotic Agents. XI, 191 pages. 2002. (Subseries LNAI).

Vol. 2555: E.-P. Lim, S. Foo, C. Khoo, H. Chen, E. Fox, S. Urs, T. Costantino (Eds.), Digital Libraries: People, Knowledge, and Technology. Proceedings, 2002. XVII, 535 pages. 2002.

Vol. 2556: M. Agrawal, A. Seth (Eds.), FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science. Proceedings, 2002. XI, 361 pages. 2002.

Vol. 2557: B. McKay, J. Slaney (Eds.), AI 2002: Advances in Artificial Intelligence. Proceedings, 2002. XV, 730 pages. 2002. (Subseries LNAI).

Vol. 2558: P. Perner, Data Mining on Multimedia Data. X, 131 pages. 2002.

Vol. 2559: M. Oivo, S. Komi-Sirviö (Eds.), Product Focused Software Process Improvement. Proceedings, 2002. XV, 646 pages. 2002.

Vol. 2560: S. Goronzy, Robust Adaptation to Non-Native Accents in Automatic Speech Recognition. Proceedings, 2002. XI, 144 pages. 2002. (Subseries LNAI).

Vol. 2561: H.C.M. de Swart (Ed.), Relational Methods in Computer Science. Proceedings, 2001. X, 315 pages. 2002.

Vol. 2562: V. Dahl, P. Wadler (Eds.), Practical Aspects of Declarative Languages. Proceedings, 2003. X, 315 pages. 2002.

Vol. 2566: T.Æ. Mogensen, D.A. Schmidt, I.H. Sudborough (Eds.), The Essence of Computation. XIV, 473 pages. 2002.

Vol. 2567: Y.G. Desmedt (Ed.), Public Key Cryptography – PKC 2003. Proceedings, 2003. XI, 365 pages. 2002.

Vol. 2568: M. Hagiya, A. Ohuchi (Eds.), DNA Computing. Proceedings, 2002. XI, 338 pages. 2003.

Vol. 2569: D. Gollmann, G. Karjoth, M. Waidner (Eds.), Computer Security – ESORICS 2002. Proceedings, 2002. XIII, 648 pages. 2002. (Subseries LNAI).

Vol. 2570: M. Jünger, G. Reinelt, G. Rinaldi (Eds.), Combinatorial Optimization – Eureka, You Shrink!. Proceedings, 2001. X, 209 pages. 2003.

Vol. 2571: S.K. Das, S. Bhattacharya (Eds.), Distributed Computing. Proceedings, 2002. XIV, 354 pages. 2002.

Vol. 2572: D. Calvanese, M. Lenzerini, R. Motwani (Eds.), Database Theory – ICDT 2003. Proceedings, 2003. XI, 455 pages. 2002.

Vol. 2574: M.-S. Chen, P.K. Chrysanthis, M. Sloman, A. Zaslavsky (Eds.), Mobile Data Management. Proceedings, 2003. XII, 414 pages. 2003.

Vol. 2575: L.D. Zuck, P.C. Attie, A. Cortesi, S. Mukhopadhyay (Eds.), Verification, Model Checking, and Abstract Interpretation. Proceedings, 2003. XI, 325 pages. 2003.

Vol. 2576: S. Cimato, C. Galdi, G. Persiano (Eds.), Security in Communication Networks. Proceedings, 2002. IX, 365 pages. 2003.

Vol. 2578: F.A.P. Petitcolas (Ed.), Information Hiding. Proceedings, 2002. IX, 427 pages. 2003.

Vol. 2580: H. Erdogmus, T. Weng (Eds.), COTS-Based Software Systems. Proceedings, 2003. XVIII, 261 pages. 2003.

Vol. 2581: J.S. Sichman, F. Bousquet, P. Davidsson (Eds.), Multi-Agent-Based Simulation II. Proceedings, 2002. X, 195 pages. 2003. (Subseries LNAI).

Vol. 2583: S. Matwin, C. Sammut (Eds.), Inductive Logic Programming. Proceedings, 2002. X, 351 pages. 2003. (Subseries LNAI).

Vol. 2588: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. Proceedings, 2003. XV, 648 pages. 2003.

Vol. 2589: E. Börger, A. Gargantini, E. Riccobene (Eds.), Abstract State Machines 2003. Proceedings, 2003. XI, 427 pages. 2003.

Vol. 2594: A. Asperti, B. Buchberger, J.H. Davenport (Eds.), Mathematical Knowledge Management. Proceedings, 2003. X, 225 pages. 2003.

Vol. 2598: R. Klein, H.-W. Six, L. Wegner (Eds.), Computer Science in Perspective. X, 357 pages. 2003.

Vol. 2600: S. Mendelson, A.J. Smola, Advanced Lectures on Machine Learning. Proceedings, 2002. IX, 259 pages. 2003. (Subseries LNAI).

Vol. 2601: M. Ajmone Marsan, G. Corazza, M. Listanti, A. Roveri (Eds.) Quality of Service in Multiservice IP Networks. Proceedings, 2003. XV, 759 pages. 2003.

Vol. 2602: C. Priami (Ed.), Computational Methods in Systems Biology. Proceedings, 2003. IX, 214 pages. 2003.

Vol. 2607: H. Alt, M. Habib (Eds.), STACS 2003. Proceedings, 2003. XVII, 700 pages. 2003.