

Hideki Imai Yuliang Zheng (Eds.)

Public Key Cryptography

Third International Workshop on Practice
and Theory in Public Key Cryptosystems, PKC 2000
Melbourne, Victoria, Australia, January 2000
Proceedings



Springer

Lecture Notes in Computer Science 1751
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Hideki Imai Yuliang Zheng (Eds.)

Public Key Cryptography

Third International Workshop
on Practice and Theory in
Public Key Cryptosystems, PKC 2000
Melbourne, Victoria, Australia, January 18-20, 2000
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Hideki Imai
University of Tokyo, Institute of Industrial Science, The Third Department
7-22-1, Roppongi, Minato-ku, Tokyo, 106-8558, Japan
E-mail: imai@iis.u-tokyo.ac.jp

Yuliang Zheng
Monash University, School of Computing and Information Technology
McMahons Road, Frankston, Melbourne, VIC 3199, Australia
E-mail: yzheng@fcit.monash.edu.au

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Public key cryptography : proceedings / Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000, Melbourne, Victoria, Australia, January 18 - 20, 2000. Hideki Imai ; Yuliang Zheng (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1751)
ISBN 3-540-66967-1

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

ISSN 0302-9743
ISBN 3-540-66967-1 Springer-Verlag Berlin Heidelberg New York

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

© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author
SPIN: 10719554 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

Preface

The PKC2000 conference was held at the Melbourne Exhibition Centre, Victoria, Australia, January 18-20, 2000. It was the third conference in the international workshop series dedicated to practice and theory in public key cryptography.

The program committee of the conference received 70 full submissions from around the world, of which 31 were selected for presentation. All submissions were reviewed by experts in the relevant areas.

The program committee consisted of 19 experts in cryptography and data security drawn from the international research community, these being Chin-Chen Chang (National Chung Cheng University, Taiwan), Claude Crépeau (McGill University, Canada), Ed Dawson (Queensland University of Technology, Australia), Yvo Desmedt (Florida State University, USA), Hideki Imai (Co-chair, University of Tokyo, Japan), Markus Jakobsson (Bell Labs, USA), Kwangjo Kim (Information and Communications University, Korea), Arjen Lenstra (Citibank, USA), Tsutomu Matsumoto (Yokohama National University, Japan), David Naccache (Gemplus, France), Eiji Okamoto (University of Wisconsin-Milwaukee, USA), Tatsuaki Okamoto (NTT Labs, Japan), Josef Pieprzyk (University of Wollongong, Australia), Jean-Jacques Quisquater (Université Catholique de Louvain, Belgium), Nigel Smart (HP Labs Bristol, UK), Vijay Varadharajan (University of Western Sydney, Australia), Serge Vaudenay (Ecole Polytechnique Fédérale de Lausanne, Switzerland), Moti Yung (CertCo, USA), and Yuliang Zheng (Co-chair, Monash University, Australia). Members of the committee spent numerous hours in reviewing the submissions and providing advice and comments on the selection of papers.

The program committee also asked expert advice of many of their colleagues, including: Masayuki Abe, Kazumaro Aoki, Paul Ashley, Joonsang Baek, Olivier Baudron, Christophe Bidan, Dan Boneh, Colin Boyd, Chris Charnes, Jean-Sébastien Coron, Ed Dawson, Paul Dumais, Kenneth Finlayson, Pierre-Alain Fouque, Atsushi Fujioka, Chandana Gamage, Juan Garay, Hossein Ghodosi, Pierre Girard, Jean-Luc Giraud, Louis Granboulan, Marc Gysin, Stuart Haber, Helena Handschuh, Ari Juels, Tetsutaro Kobayashi, Byongcheon Lee, Wei-Bin Lee, Phil MacKenzie, Wenbo Mao, William Millan, David M’Raihi, Yi Mu, Shinichi Nakahara, Kenny Nguyen, Phong Nguyen, David Pointcheval, Pascal Paillier, Ludovic Rousseau, Selwyn Russell, David Soldera, Stuart Stubblebine, Koutarou Suzuki, Christophe Tymen, Shigenori Uchiyama, Susanne Wetzel, Stefan Wolf, and Chuan-Kun Wu.

We would like to take this opportunity to thank all the program committee members and external experts for their invaluable help in producing such a high quality program. We are especially indebted to Chin-Chen Chang who made sure all the submissions assigned to him were properly reviewed in spite of the devastating earthquake and its countless aftershocks that rocked Taiwan in late September 1999.

The conference would not have been successful without the financial support from both Imai Laboratory (imailab-www.iis.u-tokyo.ac.jp) of the Institute of Industrial Science, University of Tokyo, and LINKS – Laboratory for Information and Network Security (www.pscit.monash.edu.au/links/) of the Faculty of Information Technology, Monash University. Our appreciation also goes to members of LINKS, including Jerald Chong, Chandana Gamage, Lionnel Heng, Khaled Khan, Jussi Leiwo, Patrik Mihailescu, and E-Yang Tang for their skillful and professional assistance in organizing this conference. Chandana Gamage deserves special thanks for helping out during the entire refereeing and editing process.

Last, but not least, we would like to thank all the authors who submitted their papers to the conference (including those whose submissions were not successful), as well as the conference participants from around the world, for their support which made this conference possible.

January 2000

Hideki Imai
Yuliang Zheng

PKC2000

2000 International Workshop on Practice and Theory in Public Key Cryptography

Melbourne Exhibition Centre, Australia

January 18-20, 2000

Sponsored by

Imai Laboratory of the Institute of Industrial Science,
University of Tokyo, Japan (imailab-www.iis.u-tokyo.ac.jp)
and

LINKS – Laboratory for Information and Network Security
of Monash University, Australia (www.pscit.monash.edu.au/links/)

Program Committee

Hideki Imai, Co-chair	(University of Tokyo, Japan)
Yuliang Zheng, Co-chair	(Monash University, Australia)
Chin-Chen Chang	(National Chung Cheng University, Taiwan)
Claude Crepeau	(McGill University, Canada)
Ed Dawson	(Queensland University of Technology, Australia)
Yvo Desmedt	(Florida State University, USA)
Markus Jakobsson	(Bell Labs, USA)
Kwangjo Kim	(Information and Communications University, Korea)
Arjen Lenstra	(Citibank, USA)
Tsutomu Matsumoto	(Yokohama National University, Japan)
David Naccache	(Gemplus, France)
Eiji Okamoto	(University of Wisconsin-Milwaukee, USA)
Tatsuaki Okamoto	(NTT Labs, Japan)
Josef Pieprzyk	(University of Wollongong, Australia)
Jean-Jacques Quisquater	(Université Catholique de Louvain, Belgium)
Nigel Smart	(HP Labs Bristol, UK)
Vijay Varadharajan	(University of Western Sydney, Australia)
Serge Vaudenay	(EPFL, Switzerland)
Moti Yung	(CertCo, USA)

Contents

A Practical and Secure Fault-Tolerant Conference-Key Agreement Protocol	1
<i>Wen-Guey Tzeng (Nat Chiao Tung Uni, Taiwan)</i>	
An Efficient NICE-Schnorr-Type Signature Scheme	14
<i>Detlef Hühnlein, and Johannes Merkle (secunet, Germany)</i>	
Identification of Bad Signatures in Batches	28
<i>Jarosław Pastuszak, Dariusz Michalek (Polish Acad of Sci, Poland), Josef Pieprzyk, and Jennifer Seberry (Uni of Wollongong, Australia)</i>	
Some Remarks on a Fair Exchange Protocol	46
<i>Jianying Zhou, Robert Deng, and Feng Bao (Kent Ridge Digital Labs, Singapore)</i>	
Gaudry's Variant against C_{ab} Curves	58
<i>Seigo Arita (NEC, Japan)</i>	
An Identification Scheme Based on Sparse Polynomials	68
<i>William D. Banks, Daniel Lieman (Uni of Missouri, USA), and Igor E. Shparlinski (Macquarie Uni, Australia)</i>	
A State-Based Model for Certificate Management Systems	75
<i>Chuchang Liu, Maris A. Ozols, Marie Henderson, and Tony Cant (DSTO, Australia)</i>	
Confidence Valuation in a Public-Key Infrastructure Based on Uncertain Evidence	93
<i>Reto Kohlas, and Ueli Maurer (ETH, Switzerland)</i>	
The Composite Discrete Logarithm and Secure Authentication	113
<i>David Pointcheval (ENS, France)</i>	
Chosen-Ciphertext Security for Any One-Way Cryptosystem	129
<i>David Pointcheval (ENS, France)</i>	
Short Proofs of Knowledge for Factoring	147
<i>Guillaume Poupard, and Jacques Stern (ENS, France)</i>	
Secure and Practical Tree-Structure Signature Schemes Based on Discrete Logarithms	167
<i>X.Y.Wang (Uni of Hong Kong, and Shandong Uni, China), L.C.Hui, K.P.Chow, W.W.Tsang, C.F.Chong, and H.W.Chan (Uni of Hong Kong, China)</i>	

All-or-Nothing Transform and Remotely Keyed Encryption Protocols	178
<i>Sang Uk Shin, Weon Shin, and Kyung Hyune Rhee (PuKyong Nat Uni, Korea)</i>	
Security of Public Key Certificate Based Authentication Protocols	196
<i>Wu Wen, Takamichi Saito, and Fumio Mizoguchi (Sci Uni of Tokyo, Japan)</i>	
Efficient Implementation of Schoof's Algorithm in Case of Characteristic 2	210
<i>Tetsuya Izu, Jun Kogure, and Kazuhiro Yokoyama (Fujitsu Labs, Japan)</i>	
Key Recovery in Third Generation Wireless Communication Systems	223
<i>Juanma González Nieto (QUT, Australia), DongGook Park (QUT, Australia, and Korea Telecom), Colin Boyd, and Ed Dawson (QUT, Australia)</i>	
Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications	238
<i>Katsuyuki Okeya, Hiroyuki Kurumatani (Hitachi, Japan), and Kouichi Sakurai (Kyushu Uni, Japan)</i>	
Certificates of Recoverability with Scalable Recovery Agent Security	258
<i>Eric R. Verheul (PricewaterhouseCoopers, The Netherlands)</i>	
Design Validations for Discrete Logarithm Based Signature Schemes	276
<i>Ernest Brickell (Intel, USA), David Pointcheval (ENS, France), Serge Vaudenay (EPFL, Switzerland), and Moti Yung (CertCo, USA)</i>	
Optimally Efficient Accountable Time-Stamping	293
<i>Ahto Buldas, Helger Lipmaa (Küberneetika AS, Estonia), and Berry Schoenmakers (Eindhoven Uni of Tech, The Netherlands)</i>	
“Pseudorandom Intermixing”: A Tool for Shared Cryptography	306
<i>Yair Frankel (CertCo, USA), Philip MacKenzie (Bell Labs, USA), and Moti Yung (CertCo, USA)</i>	
RSA-Based Auto-recoverable Cryptosystems	326
<i>Adam Young (Columbia Uni, USA), and Moti Yung (CertCo, USA)</i>	
Efficient and Fresh Certification	342
<i>Irene Gassko (Bell Labs, USA), Peter S. Gemmell (Uni of New Mexico, USA), and Philip MacKenzie (Bell Labs, USA)</i>	

Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions	354
<i>Ronald Cramer (ETH, Switzerland), Ivan Damgård (Aarhus Uni, Denmark), and Philip MacKenzie (Bell Labs, USA)</i>	
Cryptographic Approaches to Privacy in Forensic DNA Databases	373
<i>Philip Bohannon, Markus Jakobsson (Bell Labs, USA), and Sukamol Srikwan (Chulalongkorn Uni, Thailand)</i>	
Making Hash Functions from Block Ciphers	
Secure and Efficient by Using Convolutional Codes	391
<i>Toru Inoue (AMSL, Japan), and Kouichi Sakurai (Kyushu Uni, Japan)</i>	
Fast Implementation of Elliptic Curve Arithmetic in $GF(p^n)$	405
<i>Chae Hoon Lim, and Hyo Sun Hwang (Future Systems, Korea)</i>	
An Auction Protocol Which Hides Bids of Losers	422
<i>Kazue Sako (NEC, Japan)</i>	
Forward Secrecy and Its Application to Future Mobile Communications Security	433
<i>DongGook Park (QUT, Australia, and Korea Telecom), Colin Boyd (QUT, Australia), and Sang-Jae Moon (Kyungpook Nat Uni, Korea)</i>	
Selecting Cryptographic Key Sizes	446
<i>Arjen K. Lenstra (Citibank, USA), and Eric R. Verheul (PricewaterhouseCoopers, The Netherlands)</i>	
A Structured ElGamal-Type Multisignature Scheme	466
<i>Mike Burmester (Royal Holloway, Uni of London, UK), Yvo Desmedt (Florida State Uni, USA), Hiroshi Doi (Okayama Uni, Japan), Masahiro Mambo (Tohoku Uni, Japan), Eiji Okamoto (Uni of Wisconsin, Milwaukee, USA), Mitsuru Tada, and Yuko Yoshifuji (JAIST, Japan)</i>	
Author Index	485

A Practical and Secure Fault-Tolerant Conference-Key Agreement Protocol*

Wen-Guey Tzeng

Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 30050
tzeng@cis.nctu.edu.tw

Abstract. When a group of people wants to communicate securely over an open network, they run a conference-key protocol to establish a common conference key K such that all their communications thereafter are encrypted with the key K . In this paper we propose a practical and provably secure fault-tolerant conference-key agreement protocol under the authenticated broadcast channel model. The adversary that attacks our protocol can be either active or passive. An active adversary (malicious participant) tries to disrupt establishment of a common conference key among the honest participants, while a passive adversary tries to learn the conference key by listening to the communication of participants. We show that a passive adversary gets no information (zero knowledge) about the conference key established by the honest participants under the assumption of a variant Diffie-Hellman decision problem. We also show that the honest participants can agree on a common conference key no matter how many participants are malicious.

1 Introduction

When a group of people wants to communicate securely over an open network, they run a conference-key protocol to establish a common conference key K such that all their communications thereafter are encrypted with the key K . The first type of conference-key protocols, called conference-key distribution, is that a chairman selects a conference key and distributes the key to the participants. The second type of conference-key protocols, called conference-key agreement, is that all participants together compute a common key without a chairman. The later one is suitable for distributed environments. Conference-key protocols are also designed for various types of network connection, such as the ring connection, the star connection, the broadcast connection, etc. The conference keys of a conference-key protocol are either pre-distributed or dynamic. The conference key is fixed for a particular group of participants in a pre-distributed conference-key protocol, while it is different for each session in a dynamic conference-key protocol. The pre-distributed conference-key protocol lacks of flexibility often.

* Research supported in part by the National Science Council, Taiwan, ROC, grant NSC-88-2213-E-009-053

In this paper we propose a practical and provably secure fault-tolerant conference-key agreement protocol under the authenticated broadcast channel model. The adversary that attacks our protocol can be either active or passive. An active adversary (malicious participant) tries to disrupt establishment of a common conference key among the honest participants, while a passive adversary tries to learn the conference key by listening to the communication of participants. We tolerate the case that a malicious participant gets the conference key since the malicious participant can simply behave properly to get the key. We show that a passive adversary gets no information (zero knowledge) about the common conference key established by the honest participants under the assumption of the Diffie-Hellman decision problem. We also show that the honest participants can agree on a common conference key no matter how many participants are malicious.

We can relax the requirement of the broadcast channel being authenticated. For an un-authenticated broadcast channel, the attack from impersonators who try to impersonate participants is possible. We shall argue that our protocol is secure in this sense. We can actually use a more sophisticated commitment scheme in the protocol to achieve rigid security against impersonators. Nevertheless, in order to keep the protocol's structure and simplicity for practicality, we design our protocol as it is now.

Computing a conference key is a special case of secure multiparty computation in which a group of people evaluate a function $f(k_1, k_2, \dots)$ securely with each person possessing a private input k_i . Therefore, it is possible to have a secure conference-key agreement protocol by the generic construction for secure multiparty computation. However, there are some distinct features in the conference-key agreement protocol. First, there are no private channels between participants, which is a general assumption in secure multiparty computation. Second, a cheater's goal in a conference-key agreement protocol is to disrupt conference-key establishment among the honest participants. This is quite different from the goal of cheaters in secure multiparty computation. Third, in multiparty computation when a cheater is found, the cheater's secret x_i , which is shared into others, is recovered by honest participants so that evaluation can proceed. In conference-key agreement, since a cheater's secret is not a necessity in computing a conference key, the cheater is simply excluded from participating when found.

There have been intensive research on conference-key protocols. For example, conference-key distribution protocols (with a chairman) have been studied in [3,9,10,16], pre-distributed conference-key protocols have been studied in [4,5,19], and conference-key agreement protocols have been studied in [14,16,17,28,29]. Most proposed protocols focus on security and message efficiency for various types of network connection. Nevertheless, they do not have the capability of fault-tolerance so that a malicious participant can easily mislead other participants to compute different keys. On the other hand, Klein et al. [15] proposed a fault-tolerant conference-key agreement protocol. However, the protocol is quite inefficient and its security is not rigidly proved. Burmester

and Desmedt [7] proposed an efficient (two-round) protocol (Protocol 3) for the broadcast channel with $f(k_1, k_2, \dots, k_n) = g^{k_1 k_2 + k_2 k_3 + \dots + k_n k_1} \bmod p$. They showed that an eavesdropper cannot compute the common conference key if the Diffie-Hellman decision problem is intractable. In contrast, an eavesdropper gets zero knowledge about the common conference key in our protocol if a variant Diffie-Hellman decision problem is hard. Their basic protocol cannot withstand the attack of malicious participants. In the modified Protocol 7 (authenticated key distribution), they used an interactive proof for authenticating sent messages and showed that the protocol is secure against impersonators. The number of rounds in the protocol is proportional to the number of participants, which is not practical in some cases.

2 Model

A user in the system is a probabilistic polynomial-time Turing machine. Each user U_i has a secret information x_i and a corresponding public information y_i . The system has a public directory that records the system's public parameters and each user's public information that can be accessed by every one. All users are connected by an authenticated broadcast network such that the messages sent on the network can be identified and cannot be altered, blocked or delayed. Therefore, every one can send and receive the message on the network without interruption. No private channel exists between users. A group of users who wants to establish a conference key is called the *set of participants*. A participant may be malicious in any way.

There are two types of adversaries. A *passive adversary* who is not a participant listens to the broadcast channel and tries to learn the conference key established by the honest participants. An *active adversary* who is a participant tries to disrupt establishment of a common conference key among the honest participants. An active adversary mainly sends "malicious" messages into the broadcast channel to fool an honest participant to believe that he has computed the same conference key as that of other honest participants, while he does not indeed. We don't care about the possibility that two or more cheating participants collaborate and result in one of them or other malicious participants not being able to compute the key. This includes the following case. A malicious participant U_i sends "malicious" messages, but all honest participants compute the same key. Another malicious participant U_j , though receiving an incorrect key, still claims that he has had received the correct key. We tolerate this case since this type of collaboration between U_i and U_j do no harm to the honest participants.

A conference-key agreement protocol is secure if it can withstand attacks of passive and active adversaries. For security against a passive adversary, we mean that the adversary alone can construct a view that is computationally indistinguishable from the real conversation occurred among the participants. For security against an active adversary, we mean that if the adversary does not

follow the protocol in *any* way, the probability that he can disrupt establishment of a common conference key among honest participants is negligible.

3 Design Principles

Our protocol is component-based, that is, our protocol uses cryptographic modules as building blocks. Component-based design has many merits. First, because of using modular design, it is easy to upgrade the components of the protocol in case that better components, in either efficiency, cost, or security, are available. Also, in case that security flaws are found in a component, we can replace the component only and need not abandon the whole established system. Second, it is easier to apply strong security analysis on the protocol. Since each component has a single security goal, we can analyze each component in the focused security features. Third, it is flexible and suitable for use in a large system. A conference may be called among the participants all over the world. Flexibility of component-based design allows each user choose adequate components for each conference session. Therefore, component-based design is suitable for large and heterogeneous systems.

Suppose that each participant U_i holds a secret x_i , $1 \leq i \leq n$. They need evaluate a function f to get the conference key $K = f(k_1, k_2, \dots, k_n)$, where k_i is a randomly selected secret (sub-key) of U_i for the conference session. In our protocol, we let each participant U_i handle a function f_i and the conference-key function is

$$f(k_1, k_2, \dots, k_n) = \sum f_i(k_1, k_2, \dots, k_n)$$

where $f_i(k_1, k_2, \dots, k_n) = k_i$. Since the result k_i of f_i is independent of other parameters $k_j, j \neq i$, participant U_i can broadcast messages so that other participants can evaluate f_i in a secure multiparty computation way. As mentioned previously, our protocol is component-based. It contains the following components:

1. Component of secure multiparty computation for f_i :
2. Component of k_i commitment and verification:

Our conference key agreement protocol has the following four stages.

1. **Secret distribution and commitment:** Using the paradigm of secure multiparty computation, each participant U_i broadcasts \mathbf{w}_i so that any participant U_j can compute $f_i(\dots, k_i, \dots) = k_i$ from \mathbf{w}_i and his secret x_j . Since the computation is secure, no passive adversary shall get any information about k_i . Also, U_i broadcasts \mathbf{c}_i that commits to k_i so that other participants can verify correctness of k_i ..
2. **Sub-key computation and verification:** U_i computes k'_j of all other participants $U_j, j \neq i$. When U_i gets k'_j , he can use \mathbf{u}_j to check whether k'_j is correct.

3. **Fault detection:** If the above verification is not correct, U_i asks U_j to reveal information about commitment \mathbf{c}_i and messages \mathbf{w}_i so that all participants can determine whether U_i is cheating. If U_i detects a cheater, he deletes the cheater from his participant set and restarts the protocol.
4. **Conference-key computation:** When no faults are detected, add all sub-keys together to get the conference key.

Actually, each participant U_i can use a different method for securely computing f_i and committing to k_i as long as the methods are known by other participants.

4 A Concrete Protocol

The system has public parameters:

- p : a large prime number that is $2q + 1$, where q is a large prime also.
- H : a one-way permutation from Z_q to Z_q .
- g : a generator (primitive root) for the subgroup H_q of quadratic residues of Z_p^* .

Each user U_i has two parameters:

- Private parameter x_i : a number in Z_q^* .
- Public parameter $y_i = g^{x_i} \bmod p$. Since q is prime, y_i is a generator for H_q .

The protocol starts with that an initiator calls for a conference for a set U of participants. Without loss of generality, let $U = \{U_1, U_2, \dots, U_n\}$ be the initial participant set. Each U_i , $1 \leq i \leq n$, knows U .

1. **Secret distribution and commitment:** each participant U_i does the following:
 - (a) Randomly select $R_i, K_i \in Z_q, S_i \in Z_q^*$.
 - (b) Compute a polynomial $h_i(x)$ (over Z_q) of degree n that passes points $(j, y_j^{R_i} \bmod p \bmod q)$, $1 \leq j \leq n$, and $(0, K_i)$.
 - (c) Compute and broadcast

$$w_{ij} = h_i(n + j) \bmod q, 1 \leq j \leq n,$$

$$\alpha_i = g^{R_i} \bmod p,$$

$$\gamma_i = g^{S_i} \bmod p,$$

$$\delta_i = S_i^{-1}(H(K_i) - \gamma_i x_i) \bmod q.$$

2. **Sub-key computation and verification:** each participant U_i does the following for $j \neq i$:
 - (a) On receiving w_{jl} , $1 \leq l \leq n$, and α_j , compute polynomial $h'_j(x)$ (over Z_q) of degree n that passes $(n + l, w_{jl})$, $1 \leq l \leq n$, and $(i, \alpha_j^{x_i} \bmod p \bmod q)$.
 - (b) Let $K'_j = h'_j(0) \bmod q$.

- (c) Check whether (γ_j, δ_j) is the ElGamal signature of $H(K'_j)$ by U_j , i.e., check whether $g^{H(K'_j)} \bmod p = y_j^{\gamma_j} \gamma_j^{\delta_j} \bmod p$. If so, broadcast V_{ij} ="success". Otherwise, broadcast V_{ij} ="failure".
3. **Fault detection:** each participant U_i does the following for $j \neq i$:
- (a) On receiving V_{ji} ="failure" for some U_j : U_j claims that U_i itself is faulty.
 - i. Output R_i, K_i, S_i .
 - (b) On receiving V_{jm} ="failure": U_j claims that $U_m, m \neq i$, is faulty.
 - i. Wait for U_m 's fault detection messages R_m, K_m, S_m .
 - ii. If U_m 's fault detection messages are not received, set U_m as a malicious participant.
 - iii. On receiving R_m, K_m, S_m , check whether $w_{ml}, 1 \leq m \leq n, \alpha_m, \gamma_m$, and δ_m are correct, i.e., check whether $\alpha_m = g^{R_m} \bmod p$, whether there is an n -degree polynomial over Z_q passing points $(0, K_m)$, $(l, y_l^{R_m} \bmod p \bmod q)$, and $(n + l, w_{ml})$, $1 \leq l \leq n$, and whether (γ_m, δ_m) is the ElGamal signature of U_m on $H(K_m)$. If so, set U_j as a malicious participant. Otherwise, set U_m as a malicious participant.
 - (c) Restart the protocol by deleting malicious participants from his participant set U.
4. **Conference-key computation:** If no faults are detected in the fault detection stage, each participant U_i computes the conference

$$K = (K'_{i_1} + K'_{i_2} + \cdots + K'_{i_m}) \bmod q$$

where the current participant set is $U' = \{U_{i_1}, U_{i_2}, \dots, U_{i_m}\}$.

5 Security Analysis

We show security of the above protocol in correctness, fault tolerance and withstanding the attack of passive adversaries.

5.1 Correctness and Fault Tolerance

For correctness (completeness) of our protocol, we show that if all participants follow the protocol, they compute a common conference key.

Theorem 1 (Correctness). *If all participants follow the protocol, they compute a common conference key.*

Proof. From the broadcast messages of participant U_j , participant U_i can compute the polynomial $h_j(x) \bmod q$ passing points $(n + l, w_{jl}), 1 \leq l \leq n$, and $(i, \alpha_j^{x_i} \bmod p \bmod q)$. U_i then computes $K_j = h_j(0) \bmod q$. By the verification messages γ_j and δ_j , U_i can check whether K_j is correct. Since for fixed γ_j and δ_j the signed text $K_j \in Z_q$ is unique, all participants compute the same K_j . Thus, they compute the same conference key $K = (K_1 + K_2 + \cdots + K_n) \bmod q$.

For fault-tolerance (robustness), we show two things:

1. Any malicious participant U_i who tries to cheat honest participants to accept different K_i will be excluded from the participant sets of all honest participants.
2. An honest participant will not be excluded from the participant set of any other honest participant.

Note that it does not matter that a malicious U_i causes other malicious participants to compute different K_i .

Lemma 1. *Any malicious participant U_i who tries to cheat honest participants to accept different K_i shall be excluded from the participant sets of all honest participants.*

Proof. Malicious participants can deviate from the protocol in two ways. First, a malicious participant U_i sends "wrong" w_{il} , $1 \leq l \leq n$, α_i , γ_i and δ_i so that two honest participants U_j and U_m compute different K_i . In this case, one of them, say U_j , shall send $V_{ji} =$ "failure" since γ_i and δ_i can not be the ElGamal signature of two different K_i 's. Then, U_i has to broadcast R_i , K_i and S_i for verification. Every honest participant verifies whether $\alpha_i = g^{R_i} \bmod p$, (γ_i, δ_i) is the signature of $H(K_i)$, and the polynomial passing $(n+l, w_{il})$, $1 \leq l \leq n$ and $(0, K_i)$ also passes points $(j, y_j^{R_i} \bmod p \bmod q)$, $1 \leq j \leq n$. Since the honest U_j claims that K_i is wrong, for all participants at least one of the above checkings cannot hold. Therefore, all honest participants exclude U_i from their participant sets.

Second, U_i sends $V_{ij} =$ "failure" of claiming that U_j is malicious, while U_j is indeed honest. In this case, U_j broadcasts R_j , K_j and S_j to prove his honesty. Since U_j is honest, all honest participants decide that U_i is malicious. Therefore, the malicious U_i is excluded by all honest participants.

Lemma 2. *No honest participant excludes any other honest participant from his participant set.*

Proof. Since an honest participant U_i follows the protocol, his broadcast messages make all participants compute the same K_i . Even some malicious participant U_j claims that he is faulty, he can send R_i , K_i and S_i to prove his honesty. Therefore, no honest participant shall exclude U_i from his participant set.

By the above two lemmas, we can show that all honest participants compute the same conference key even the majority of the participants are malicious.

Theorem 2 (Robustness). *All honest participants have the same participant set and thus they compute same conference key no matter how many participants are malicious.*

Proof. By the above two lemmas, each honest participant's participant set consists of two types of participants: honest participants and those participants U_i , though deviating from the protocol, make all honest participants compute the same K_i . Therefore, all honest participants compute the same conference key.

5.2 Security against Passive Attackers

A passive attacker (eavesdropper) tries to learn information about the conference key by listening the broadcast channel. We show that an eavesdropper cannot get any information about K_i of U_i . Since each participant chooses his K_i independently, we show that the attacker's view of the messages broadcast by U_i on the broadcast channel can be simulated without knowing the secrets x_i and K_i .

We need an assumption to show that the simulated transcript is computationally indistinguishable from the real one. This assumption is a little stronger than that about the regular Diffie-Hellman decision problem that is discussed in some papers [6, 21, 27]. The assumption about the regular Diffie-Hellman decision problem is that for any given $y_1, y_2 \in H_q - \{1\}$ and $u_1, u_2 \in H_q$,

$$(y_1, y_2, y_1^R \bmod p, y_2^R \bmod p)$$

and

$$(y_1, y_2, u_1, u_2)$$

are computationally indistinguishable and thus

$$(y_1, y_2, y_1^R \bmod p \bmod q, y_2^R \bmod p \bmod q)$$

and

$$(y_1, y_2, u_1 \bmod q, u_2 \bmod q)$$

are computationally indistinguishable. Note that y_1 and y_2 must be quadratic residues of Z_p^* , otherwise one can tell apart the above probability distributions. We note that $u_1 \bmod q$ and $u_2 \bmod q$ do not range all over Z_q . Therefore, we need an assumption about a variation of the Diffie-Hellman decision problem.

Assumption 1 (Variant Diffie-Hellman decision problem). Let $p = 2q + 1$ and H_q be the quadratic-residue subgroup of Z_p^* . Given any generators $y_1, y_2 \in H_q - \{1\}$, the following two random-variable tuples are computationally indistinguishable:

$$(y_1, y_2, y_1^R \bmod p \bmod q, y_2^R \bmod p \bmod q)$$

and

$$(y_1, y_2, u_1, u_2).$$

where $R, u_1, u_2 \in Z_q$.

The simulator of the adversary's view on broadcast messages of U_i does the following:

1. Randomly select $w'_{ij} \in Z_q, 1 \leq j \leq n, R'_i \in Z_q, S'_i \in Z_q^*, \delta'_i \in Z_q$,
2. Output the simulated transcript:

$$w'_{ij}, 1 \leq j \leq n,$$

$$\alpha'_i = g^{R'_i} \bmod p,$$

$$\gamma'_i = g^{S'_i} \bmod p,$$

$$\delta'_i.$$

We now show, on random variables $K_i, R_i \in Z_q, S_i \in Z_q^*$, the real view

$$(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i, \gamma_i, \delta_i)$$

and, on random variables $w'_{ij} \in Z_q, 1 \leq j \leq n, R'_i \in Z_q, S'_i \in Z_q^*, \delta'_i \in Z_q$, the simulated view

$$w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i, \gamma'_i, \delta'_i$$

are computationally indistinguishable, where $\alpha_i = g^{R_i} \bmod p$, $\gamma_i = g^{S_i} \bmod p$, $\delta_i = S_i^{-1}(K_i - \gamma_i x_i) \bmod q$, $\alpha'_i = g^{R'_i} \bmod p$, $\gamma'_i = g^{S'_i} \bmod q$, and $w_{ij} = h_i(n+j), 1 \leq j \leq n$, is described in our protocol. Since for any $\gamma_0 \in H_q - \{1\}$ and $\delta_0 \in Z_q$,

$$\Pr[\gamma_i = \gamma_0, \delta_i = \delta_0] = \Pr[\gamma'_i = \gamma_0, \delta'_i = \delta_0] = \frac{1}{q(q-1)},$$

we only have to consider the probability distributions

$$\Pr[(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i)] | \gamma_i = \gamma_0, \delta_i = \delta_0]$$

and

$$\Pr[(w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i)] | \gamma'_i = \gamma_0, \delta'_i = \delta_0].$$

For any fixed γ_0 and δ_0 , the random variable K_i is fixed, say k_0 . We have

$$\begin{aligned} & \Pr[(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i)] | \gamma_i = \gamma_0, \delta_i = \delta_0 \\ &= \Pr[(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i)] | K_i = k_0 \end{aligned}$$

and

$$\begin{aligned} & \Pr[(w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i)] | \gamma_0, \delta_0 \\ &= \Pr[(w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i)] \end{aligned}$$

for some $k_0 \in Z_q$. We show that they are computationally indistinguishable.

Lemma 3. *Under Assumption 1, for any fixed $K = k_0$, on random variables $R_i, R'_i, w'_{i1}, w'_{i2}, \dots, w'_{in} \in Z_q$,*

$$(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i)$$

and

$$(w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i)$$

are computationally indistinguishable.

Proof. By the assumption

$$(y_1, y_2, \dots, y_n, y_1^{R_i} \bmod p \bmod q, y_2^{R_i} \bmod p \bmod q, \dots, y_n^{R_i} \bmod p \bmod q,$$

$$g^{R_i} \bmod p)$$

and

$$(y_1, y_2, \dots, y_n, u_1, u_2, \dots, u_n, g^{R'_i} \bmod p)$$

are computationally indistinguishable, where $R_i, R'_i, u_j \in Z_q, 1 \leq j \leq n$. Let $K = k_0$ be fixed. Let \bar{h}_i (over Z_q) be the n -degree polynomial passing points $(0, k_0)$ and $(j, u_j), 1 \leq j \leq n$. By applying a polynomial interpolation on them, we have that,

$$(w_{i1}, w_{i2}, \dots, w_{in}, g^{R_i} \bmod p)$$

and

$$(\bar{w}_{i1}, \bar{w}_{i2}, \dots, \bar{w}_{in}, g^{R'_i} \bmod p)$$

are computationally indistinguishable, where $\bar{w}_{ij} = \bar{h}_i(n + j) \bmod q, 1 \leq j \leq n$. Since for any $\bar{w}_{ij}^0 \in Z_q, 1 \leq j \leq n$, and $\bar{\alpha}_0 \in H_q$,

$$\Pr[(\bar{w}_{i1}, \bar{w}_{i2}, \dots, \bar{w}_{in}, g^{R'} \bmod p) = (\bar{w}_{i1}^0, \bar{w}_{i2}^0, \dots, \bar{w}_{in}^0, \bar{\alpha}_0)] = \frac{1}{q^{n+1}},$$

thus $(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i)$ and $(w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i)$ are computationally indistinguishable.

Therefore, the simulator outputs a transcript that is computationally indistinguishable from the real one.

Theorem 3 (Privacy). *Under Assumption 1, for any $i, 1 \leq i \leq n$, the real communication transcript of U_i*

$$(w_{i1}, w_{i2}, \dots, w_{in}, \alpha_i, \gamma_i, \delta_i)$$

and the simulated one

$$(w'_{i1}, w'_{i2}, \dots, w'_{in}, \alpha'_i, \gamma'_i, \delta'_i)$$

are computationally indistinguishable, where random variables $R_i, K_i \in Z_q, S_i \in Z_q^*$ and $w'_{i1}, w'_{i2}, \dots, w'_{in} \in Z_q, S'_i \in Z_q^*, \delta'_i \in Z_q$.

Proof. This is obvious by Lemma 3.

6 Security against Impersonators

We have discussed the security of our protocol against eavesdroppers and malicious participants. We have assumed that the broadcast channel is authentic. Therefore, no impersonator (outsider) can succeed in pretending to be a legal participant without being detected. It is easy to enforce "authentication" on the broadcast channel since we can simply require participants to sign their broadcast messages. However, it is cumbersome.

In fact, we can relax the requirement of "authenticated" broadcast channel. In our protocol, we require the participant U_i to sign $H(K_i)$, instead of K_i . The reason is to prevent impersonation of U_i . We note that an outsider without knowing U_i 's secret x_i can sign a random message $m = -\gamma_i ab^{-1} \bmod p$ by choosing $\gamma_i = g^a y_i^b \bmod p$ and $\delta_i = -\gamma_i b^{-1} \bmod q$ first for $a \in Z_q$ and $b \in$

Z_q^* [20]. If we only require U_i to sign $K_i = m$, the impersonator can share K_i with other participants even though he cannot compute other participants' K_j 's.

We don't have a rigid proof for our protocol's strength against impersonators. Nevertheless, we give some explanation. First, if the impersonator chooses K_i first and then signs $H(K_i)$, he has to sign a chosen message $H(K_i)$, which is not known to be possible in the ElGamal signature scheme. Second, if the impersonator chooses $m = H(K_i)$ first, he has to compute $K_i = H^{-1}(m)$ in order to share K_i with other participants. This occurs with only a negligible probability under H being a one-way permutation. A strong evidence shows that this approach (full-domain-hash-then-sign) is secure against signature forgery, thus impersonators [1].

7 Conclusion

Assuming an authenticated broadcast channel, we have presented a conference-key agreement protocol that is provably secure against passive and active adversaries under the assumption of a variant Diffie-Hellman decision problem. We argue that our protocol is secure against active impersonators if the full-domain-hash-then-sign paradigm for ElGamal signature is secure.

Our protocol is round-efficient. It uses only two rounds to compute a common conference key after all malicious participants are detected. Nevertheless, the size of messages that each participant sends is proportional to the number of participants. It is interesting to design a provably secure conference-key agreement protocol with both round- and message-efficiency.

References

1. M. Bellare, P. Rogaway, "The Exact Security of Digital Signatures, How to Sign with RSA and Rabin", Advances in Cryptology: Proceedings of Eurocrypt '96, Lecture Notes in Computer Science 1070, Springer-Verlag, pp.399-416, 1996 [11](#)
2. M. Ben-Or, S. Goldwasser, A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation", In Proceedings of the 20th ACM Symposium on the Theory of Computing, pp.1-10, 1988.
3. S. Berkovits, "How to Broadcast a Secret", Advances in Cryptology: Proceedings of Eurocrypt '91, Lecture Notes in Computer Science 547, Springer-Verlag, pp.535-541, 1991. [2](#)
4. R. Blom, "An Optimal Class of Symmetric Key Generation Systems", Advances in Cryptology: Proceedings of Eurocrypt '84, Lecture Notes in Computer Science 196, Springer-Verlag, pp.335-338, 1985. [2](#)
5. C. Blundo, A.D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences", Advances in Cryptology: Proceedings of Crypto '92, Lecture Notes in Computer Science 740, Springer-Verlag, pp.471-486, 1993. [2](#)
6. D. Boneh, R. Venkatesan, "Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Problems", Advances in Cryptology: Proceedings of Crypto '96, Lecture Notes in Computer Science 1109, Springer-Verlag, pp.129-142, 1996. [8](#)

7. M. Burmester, Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", Advances in Cryptology: Proceedings of Eurocrypt '94, Lecture Notes in Computer Science 950, Springer-Verlag, pp.275-286, 1995. 3
8. R. Canetti, A. Herzberg, "Maintaining Security in the Presence of Transient Faults", Advances in Cryptology: Proceedings of Crypto '94, Lecture Notes in Computer Science 839, Springer-Verlag, pp.425-438, 1994.
9. C.C. Chang, C.H. Lin, "How to Converse Securely in a Conference", In Proceedings of IEEE Security Technology, 30th Annual 1996 International Carnahan Conference, pp.42-45, 1996. 2
10. C.C. Chang, T.C. Wu, C.P. Chen, "The Design of a Conference Key Distribution System", Advances in Cryptology: Proceedings of Auscrypt '92, Lecture Notes in Computer Science 718, Springer-Verlag, pp.459-466, 1992. 2
11. W. Diffie, M. Hellman, "New Directions in Cryptography", IEEE Transaction of Information Theory, Vol. IT-22, pp.644-654, 1976.
12. M. Fitzi, M. Hirt, U. Maurer, "Trading Correctness for Privacy in Unconditional Multi-Party Computation", Advances in Cryptology: Proceedings of Crypto '98, Lecture Notes in Computer Science 1462, Springer-Verlag, pp.121-136, 1998.
13. T. Hwang, J.L. Chen, "Identity-Based Conference Key Broadcast Systems", IEE Computers and Digital Techniques, Vol. 141, No. 1, pp.57-60, 1994.
14. I. Ingemarsson, D.T. Tang, C.K. Wong, "A Conference Key Distribution System", IEEE Transactions on Information Theory, Vol. IT-28, No. 5, pp.714-720, 1982. 2
15. B. Klein, M. Otten, T. Beth, "Conference Key Distribution Protocols in Distributed Systems", In Proceedings of Codes and Ciphers-Cryptography and Coding IV, IMA, pp.225-242, 1995. 2
16. K. Koyama, "Secure Conference Key Distribution Schemes for Conspiracy Attack", Advances in Cryptology: Proceedings of Eurocrypt '92, Lecture Notes in Computer Science 658, Springer-Verlag, pp.449-453, 1993. 2
17. K. Koyama, K. Ohta, "Identity-Based Conference Key Distribution Systems", Advances in Cryptology: Proceedings of Crypto '87, Lecture Notes in Computer Science 293, Springer-Verlag, pp.175-184, 1988. 2
18. K. Koyama, K. Ohta, "Security of Improved Identity-Based Conference Key Distribution Systems", Advances in Cryptology: Proceedings of Eurocrypt '88, Lecture Notes in Computer Science 330, Springer-Verlag, pp.11-19, 1988.
19. T. Matsumoto, H. Imai, "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem", Advances in Cryptology: Proceedings of Crypto '87, Lecture Notes in Computer Science 293, Springer-Verlag, pp.185-193, 1988. 2
20. C.J. Mitchell, F. Piper, P. Wild, "Digital Signature", In Contemporary Cryptography, The Science of Information Integrity, pp.325-378, IEEE Press, 1992. 11
21. M. Naor, O. Reingold, "Number-theoretic Constructions of Efficient Pseudorandom Functions", In Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, 1997. 8
22. R. Ostrovsky, M. Yung, "How to Withstand Mobile Virus Attacks", In Proceedings of ACM Symposium on Principles of Distributed Computing, pp.51-61, 1991.
23. T. Rabin, M. Ben-Or, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority", Proceedings of the 26th ACM Symposium on the Theory of Computing, pp.73-85, 1989.
24. R.A. Rueppel, P.C. Van Oorschot, "Modern Key Agreement Techniques", Computer Communications, 1994.
25. A. Shamir, "How to share a secret", Communications of the ACM, Vol. 22, pp.612-613, 1979.

26. A. Shimbo, S. I. Kawamura, "Cryptanalysis of Several Conference Key Distribution Schemes", Advances in Cryptology: Proceedings of Asiacrypt '91, Lecture Notes in Computer Science 739, Springer-Verlag, pp.265-276, 1993.
27. V. Shoup, "Lower Bounds for Discrete Logarithms and Related Problems", Advances in Cryptology: Proceedings of Eurocrypt '97, Lecture Notes in Computer Science 1233, Springer-Verlag, pp.256-266, 1997. 8
28. D.G. Steer, L. Strawczynski, W. Diffie, M. Wiener, "A Secure Audio Teleconference System", Advances in Cryptology: Proceedings of Crypto '88, Lecture Notes in Computer Science 409, Springer-Verlag, pp.520-528, 1990. 2
29. T.C. Wu, "Conference Key Distribution System with User Anonymity Based on Algebraic Approach", Proceedings of IEE Computers and Digital Techniques, Vol. 144, No 2, pp.145-148, 1997. 2
30. Y. Yacobi, "Attack on the Koyama-Ohta Identity Based Key Distribution Scheme", Advances in Cryptology: Proceedings of Crypto '87, Lecture Notes in Computer Science 293, Springer-Verlag, pp.429-433, 1988.

An Efficient NICE-Schnorr-Type Signature Scheme

Detlef Hühnlein and Johannes Merkle

secunet Security Networks AG
Mergenthalerallee 77-81
D-65760 Eschborn, Germany
{huehnlein,merkle}@secunet.de

Abstract. Recently there was proposed a novel public key cryptosystem [17] based on non-maximal imaginary quadratic orders with *quadratic decryption time*. This scheme was later on called NICE for **N**ew **I**deal **C**oset **E**ncryption [6]. First implementations show that the decryption is as efficient as RSA-encryption with $e = 2^{16} + 1$. It was an open question whether it is possible to construct comparably efficient signature schemes based on non-maximal imaginary quadratic orders. The major drawbacks of the ElGamal-type [7] and RSA/Rabin-type signature schemes [8] proposed so far are the *slow signature generation* and the *very inefficient system setup*, which involves the computation of the class number $h(\Delta_1)$ of the maximal order with a subexponential time algorithm. To avoid this tedious computation it was proposed to use *totally* non-maximal orders, where $h(\Delta_1) = 1$, to set up DSA analogues. Very recently however it was shown in [10], that the discrete logarithm problem in this case can be reduced to finite fields and hence there seems to be no advantage in using DSA analogues based on totally non-maximal orders.

In this work we will introduce an efficient NICE-Schnorr-type signature scheme based on conventional non-maximal imaginary quadratic orders which solves both above problems. It gets its strength from the difficulty of *factoring* the discriminant $\Delta_p = -rp^2$, r, p prime. To avoid the computation of $h(\Delta_1)$, our proposed signature scheme only operates in (a subgroup of) the kernel of the map ϕ_{Cl}^{-1} , which allows to switch from the class group of the non-maximal order to the maximal order. Note that a similar setup is used in NICE. For an efficient signature generation one may use the novel arithmetic [9] for elements of $\text{Ker}(\phi_{Cl}^{-1})$. While the signature generation using this arithmetic is already slightly faster than in the original scheme, we will show in this work that we can even do better by applying the Chinese Remainder Theorem for $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$. First implementations show that the signature generation of our scheme is *more than twice as fast* as in the original scheme in \mathbb{F}_p^* , which makes it very attractive for practical applications.

1 Introduction

Since nobody can guarantee that currently used cryptosystems based on the difficulty of factoring or the computation of discrete logarithms in some group stay secure forever it is important to consider different primitives and groups for the construction of cryptosystems. On the other hand the continuously growing popularity of cryptosystems based on elliptic curves emphasize that certain mathematical structures seem to allow more efficient implementation for the same conjectured level of security.

Another recently proposed mathematical structure which allows the construction of very efficient cryptosystems are *non-maximal imaginary quadratic orders*. For a recent survey of cryptosystems based on quadratic orders we refer to the forthcoming [11]. For example it was shown in [17] that there is a public key cryptosystem which has *quadratic decryption time*. To our knowledge this is the only scheme having this property. First implementations show that the decryption is about as efficient as the encryption with RSA with $e = 2^{16} + 1$. Note that this is a very important feature, as the decryption often takes place in a device with limited computational power, such as a smart card. It was an open question whether there is also an efficient signature scheme based on these non-maximal imaginary quadratic orders. All currently proposed signature schemes based on this structure have different drawbacks: The signature generation of the ElGamal analogue [7] and the RSA/Rabin analogues [8] based on conventional non-maximal orders is fairly inefficient. In fact, except from the Rabin-analogue, one uses the particular structure of the *non-maximal* order only to *set up* the system. The signature generation itself has to be performed in the public non-maximal order, which does not allow very efficient computation. For the system setup one has to compute the class number $h(\Delta_1)$ of the maximal order, where $|\Delta_1| > 2^{200}$ to prevent Δ_p from being factored using the Elliptic Curve Method (ECM). The computation of $h(\Delta_1)$ is done with an analogue of the quadratic sieve with subexponential running time and hence is very inefficient. To avoid this computation it was proposed in [8] to use totally non-maximal orders, where $h(\Delta_1) = 1$. Using the recently developed exponentiation technique [9] one is able to implement DSA analogues in these totally non-maximal orders almost as efficiently as conventional DSA in \mathbb{F}_p^* for the same *conjectured* level of security. However, even more recently, it was shown in [10] that discrete logarithms in the class group of *totally* non-maximal imaginary quadratic orders $Cl(\Delta_p)$ can be reduced to discrete logarithms in finite fields and hence there seems to be no advantage in using this DSA analogue.

In this work we will introduce an efficient NICE-Schnorr-type signature scheme based on non-maximal imaginary quadratic orders which solves both above problems:

At first the *system-setup* is very fast, because we do not have to compute the class number of the maximal order $h(\Delta_1)$, but only compute in (a subgroup of) the *kernel* of ϕ_{Cl}^{-1} instead, which cardinality is known in advance. This is a similar situation as for the NICE cryptosystem. As noted in [17], the restriction

to elements of the kernel does not seem to introduce any weakness as long as the conductor p is kept secret and hence our scheme is based on the difficulty of factoring $\Delta_p = \Delta_1 p^2$.

Second the *signature generation* of our proposed scheme is also very fast. To perform the exponentiation of a generator \mathbf{g} of a 160 bit subgroup of order q of $\text{Ker}(\phi_{Cl}^{-1})$ one can use the recently developed arithmetic [9], which is about *twenty* times as fast as standard ideal arithmetic. This arithmetic allows the signer to replace the fairly inefficient ideal arithmetic in the non-maximal order by computations in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$.

In this work we will show that one can even do better by application of the Chinese Remainder Theorem for $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$. If $\left(\frac{\Delta_1}{p}\right) = 1$ then there is an isomorphism $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \simeq \mathbb{F}_p^* \otimes \mathbb{F}_p^*$. Thus the signature generation in our scheme essentially consists of two exponentiations in \mathbb{F}_p^* . Considering the best algorithms (NFS and ECM) it is reasonable to assume that factoring $\Delta_p = -rp^2$, $p, r \approx 2^{340}$ prime, is "about as hard" as computing discrete logarithms in $\mathbb{F}_{p'}^*$ with p' about 1000 bits. Note that while it is *conjectured* that factoring numbers of the form rp^2 is considerably easier than factoring $n = pq$ there is only an ECM-variant [18] known which is able to make use of this special structure and if $r, p > 2^{240}$ this method is clearly infeasible. Thus the bitlength of the modulus p in our exponentiations is only *about one third* of the bitlength of the modulus in the original Schnorr scheme. Hence we end up with a signature generation which is *more than twice as fast* as for the original Schnorr scheme [20], which in turn is much faster than that of RSA for example.

Note that for possible Schnorr analogues working in subgroups of $(\mathbb{Z}/n\mathbb{Z})^*$ for composite n , one needs to be *very careful* as pointed out in [14]. This issue and the *entirely different* situation here is discussed in Section 4.

The paper is organized as follows: Section 2 will provide the necessary background and notations of non-maximal imaginary quadratic orders used in this work. In Section 3 we will explain the proposed signature scheme. In Section 4 we will consider the security of our scheme. In Section 5 we will introduce the novel exponentiation technique using CRT in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ and give timings of a first implementation. This will show that the signature generation of our scheme is more than twice as fast in the original Schnorr scheme, which makes our scheme very attractive for practical application.

2 Necessary Preliminaries and Notations of Imaginary Quadratic Orders

The basic notions of imaginary quadratic number fields may be found in [2,3]. For a more comprehensive treatment of the relationship between maximal and non-maximal orders we refer to [4,7,9,10].

Let $\Delta \equiv 0, 1 \pmod{4}$ be a negative integer, which is not a square. The quadratic order of discriminant Δ is defined to be

$$\mathcal{O}_\Delta = \mathbb{Z} + \omega \mathbb{Z},$$

where

$$\omega = \begin{cases} \sqrt{\frac{\Delta}{4}}, & \text{if } \Delta \equiv 0 \pmod{4}, \\ \frac{1+\sqrt{\Delta}}{2}, & \text{if } \Delta \equiv 1 \pmod{4}. \end{cases} \quad (1)$$

The standard representation of some $\alpha \in \mathcal{O}_\Delta$ is $\alpha = x + y\omega$, where $x, y \in \mathbb{Z}$.

If Δ_1 is squarefree, then \mathcal{O}_{Δ_1} is the *maximal order* of the quadratic number field $\mathbb{Q}(\sqrt{\Delta_1})$ and Δ_1 is called a fundamental discriminant. The *non-maximal order* of conductor $p > 1$ with (non-fundamental) discriminant $\Delta_p = \Delta_1 p^2$ is denoted by \mathcal{O}_{Δ_p} . We will always assume in this work that the conductor p is prime. Furthermore we will omit the subscripts to reference arbitrary (fundamental or non-fundamental) discriminants. Because $\mathbb{Q}(\sqrt{\Delta_1}) = \mathbb{Q}(\sqrt{\Delta_p})$ we also omit the subscripts to reference the number field $\mathbb{Q}(\sqrt{\Delta})$. The standard representation of an \mathcal{O}_Δ -ideal is

$$\mathfrak{a} = q \left(\mathbb{Z} + \frac{b + \sqrt{\Delta}}{2a} \mathbb{Z} \right) = (a, b), \quad (2)$$

where $q \in \mathbb{Q}_{>0}$, $a \in \mathbb{Z}_{>0}$, $c = (b^2 - \Delta)/(4a) \in \mathbb{Z}$, $\gcd(a, b, c) = 1$ and $-a < b \leq a$. The norm of this ideal is $\mathcal{N}(\mathfrak{a}) = aq^2$. An ideal is called primitive if $q = 1$. A primitive ideal is called *reduced* if $|b| \leq a \leq c$ and $b \geq 0$, if $a = c$ or $|b| = a$. It can be shown, that the norm of a reduced ideal \mathfrak{a} satisfies $\mathcal{N}(\mathfrak{a}) \leq \sqrt{|\Delta|}/3$ and conversely that if $\mathcal{N}(\mathfrak{a}) \leq \sqrt{|\Delta|}/4$ then the ideal \mathfrak{a} is reduced. We denote the reduction operator in the maximal order by $\rho_1()$ and write $\rho_p()$ for the reduction operator in the non-maximal order of conductor p .

The group of invertible \mathcal{O}_Δ -ideals is denoted by \mathcal{I}_Δ . Two ideals $\mathfrak{a}, \mathfrak{b}$ are equivalent, if there is a $\gamma \in \mathbb{Q}(\sqrt{\Delta})$, such that $\mathfrak{a} = \gamma \mathfrak{b}$. This equivalence relation is denoted by $\mathfrak{a} \sim \mathfrak{b}$. The set of principal \mathcal{O}_Δ -ideals, i.e. which are equivalent to \mathcal{O}_Δ , are denoted by \mathcal{P}_Δ . The factor group $\mathcal{I}_\Delta/\mathcal{P}_\Delta$ is called the *class group* of \mathcal{O}_Δ denoted by $Cl(\Delta)$. $Cl(\Delta)$ is a finite abelian group with neutral element \mathcal{O}_Δ . Algorithms for the group operation (multiplication and reduction of ideals) can be found in [3]. The order of the class group is called the *class number* of \mathcal{O}_Δ and is denoted by $h(\Delta)$.

Our cryptosystem makes use of the relation between the maximal and non-maximal orders. Any non-maximal order may be represented as $\mathcal{O}_{\Delta_p} = \mathbb{Z} + p\mathcal{O}_{\Delta_1}$. If $h(\Delta) = 1$ then \mathcal{O}_{Δ_p} is called a *totally non-maximal* imaginary quadratic order of conductor p . An \mathcal{O}_Δ -ideal \mathfrak{a} is called prime to p , if $\gcd(\mathcal{N}(\mathfrak{a}), p) = 1$. It is well known, that all \mathcal{O}_{Δ_p} -ideals prime to the conductor are invertible. In every class there is an ideal which is prime to any given number. The algorithm `FindIdealPrimeTo` in [7] will compute such an ideal. If we denote the (principal) \mathcal{O}_{Δ_p} -ideals, which are prime to p by $\mathcal{P}_{\Delta_p}(p)$ and $\mathcal{I}_{\Delta_p}(p)$ respectively then there

is an isomorphism

$$\mathcal{I}_{\Delta_p}(p)/\mathcal{P}_{\Delta_p}(p) \simeq \mathcal{I}_{\Delta_p}/\mathcal{P}_{\Delta_p} = Cl(\Delta_p). \quad (3)$$

Thus we may 'neglect' the ideals which are not prime to the conductor, if we are only interested in the class group $Cl(\Delta_p)$. There is an isomorphism between the group of \mathcal{O}_{Δ_p} -ideals which are prime to p and the group of \mathcal{O}_{Δ_1} -ideals, which are prime to p , denoted by $\mathcal{I}_{\Delta_1}(p)$ respectively:

Proposition 1. *Let \mathcal{O}_{Δ_p} be an order of conductor p in an imaginary quadratic field $\mathbb{Q}(\sqrt{\Delta})$ with maximal order \mathcal{O}_{Δ_1} .*

- (i.) *If $\mathfrak{A} \in \mathcal{I}_{\Delta_1}(p)$, then $\mathfrak{a} = \mathfrak{A} \cap \mathcal{O}_{\Delta_p} \in \mathcal{I}_{\Delta_p}(p)$ and $\mathcal{N}(\mathfrak{A}) = \mathcal{N}(\mathfrak{a})$.*
- (ii.) *If $\mathfrak{a} \in \mathcal{I}_{\Delta_p}(p)$, then $\mathfrak{A} = \mathfrak{a}\mathcal{O}_{\Delta_1} \in \mathcal{I}_{\Delta_1}(p)$ and $\mathcal{N}(\mathfrak{a}) = \mathcal{N}(\mathfrak{A})$.*
- (iii.) *The map $\varphi : \mathfrak{A} \mapsto \mathfrak{A} \cap \mathcal{O}_{\Delta_p}$ induces an isomorphism $\mathcal{I}_{\Delta_1}(p) \xrightarrow{\sim} \mathcal{I}_{\Delta_p}(p)$. The inverse of this map is $\varphi^{-1} : \mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_1}$.*

Proof: See [4, Proposition 7.20, page 144]. □

Thus we are able to switch to and from the maximal order. The algorithms `GoToMaxOrder(a, p)` to compute φ^{-1} and `GoToNonMaxOrder(A, p)` to compute φ respectively may be found in [7].

It is important to note that the isomorphism φ is between the ideal groups $\mathcal{I}_{\Delta_1}(p)$ and $\mathcal{I}_{\Delta_p}(p)$ and *not the class groups*.

If, for $\mathfrak{A}, \mathfrak{B} \in \mathcal{I}_{\Delta_1}(p)$ we have $\mathfrak{A} \sim \mathfrak{B}$, it is not necessarily true that $\varphi(\mathfrak{A}) \sim \varphi(\mathfrak{B})$.

On the other hand, equivalence *does* hold under φ^{-1} . More precisely we have the following:

Proposition 2. *The isomorphism φ^{-1} induces a surjective homomorphism $\phi_{Cl}^{-1} : Cl(\Delta_p) \rightarrow Cl(\Delta_1)$, where $\mathfrak{a} \mapsto \rho_1(\varphi^{-1}(\mathfrak{a}))$.*

Proof: This immediately follows from the short exact sequence:

$$Cl(\Delta_p) \longrightarrow Cl(\Delta_1) \longrightarrow 1$$

(see [16, Theorem 12.9, p. 82]). □

In the following we will study the kernel $\text{Ker}(\phi_{Cl}^{-1})$ of the above map ϕ_{Cl}^{-1} and hence the relation between a class in the maximal order and the associated classes in the non-maximal order in more detail. We start with yet another interpretation of the class group $Cl(\Delta_p)$.

Proposition 3. *Let \mathcal{O}_{Δ_p} be an order of conductor p in a quadratic field. Then there are natural isomorphisms*

$$Cl(\Delta_p) \simeq \mathcal{I}_{\Delta_p}(p)/\mathcal{P}_{\Delta_p}(p) \simeq \mathcal{I}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1, \mathbb{Z}}(p),$$

where $\mathcal{P}_{\Delta_1, \mathbb{Z}}(p)$ denotes the subgroup of $\mathcal{I}_{\Delta_1}(p)$ generated by the principal ideals of the form $\alpha\mathcal{O}_{\Delta_1}$ where $\alpha \in \mathcal{O}_{\Delta_1}$ satisfies $\alpha \equiv a \pmod{p\mathcal{O}_{\Delta_1}}$ for some $a \in \mathbb{Z}$ such that $\gcd(a, p) = 1$.

Proof: See [4, Proposition 7.22, page 145]. \square

The following corollary is an immediate consequence.

Corollary 1. *With notations as above we have the following isomorphism*

$$\text{Ker}(\phi_{Cl}^{-1}) \simeq \mathcal{P}_{\Delta_1}(f) / \mathcal{P}_{\Delta_1, \mathbb{Z}}(f).$$

The next result explains the relation between $\text{Ker}(\phi_{Cl}^{-1})$ and $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$.

Proposition 4. *The map $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \rightarrow \text{Ker}(\phi_{Cl}^{-1})$, where $\alpha \mapsto \varphi(\alpha\mathcal{O}_{\Delta_1})$ is a surjective homomorphism.*

Proof: This is shown in the more comprehensive proof of Theorem 7.24 in [4] (page 147). \square

Thus one may reduce the arithmetic in $\text{Ker}(\phi_{Cl}^{-1})$ to more efficient computation in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$. This is precisely what was proposed in [9]. Using the naive "generator-arithmetic" as introduced there one is able to perform an exponentiation in $\text{Ker}(\phi_{Cl}^{-1})$ about twenty times as fast as by using standard ideal arithmetic. In Section 5 we will show that one can even do much better by applying the CRT in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$.

Finally, we will give the exact relationship between the class numbers $h(\Delta_1)$ and $h(\Delta_p)$.

Proposition 5. *Let $\Delta_1 < -4$, $\Delta_1 \equiv 0, 1 \pmod{4}$ and p prime. Then $h(\Delta_p) = h(\Delta_1) \left(p - \left(\frac{\Delta_1}{p}\right)\right)$ and $|\text{Ker}(\phi_{Cl}^{-1})| = \left(p - \left(\frac{\Delta_1}{p}\right)\right)$, where $\left(\frac{\Delta_1}{p}\right)$ is the Kronecker-symbol.*

Proof: Because $\mathcal{O}_{\Delta_1}^* = \mathcal{O}_{\Delta_p}^* = \{\pm 1\}$, for $\Delta_p = \Delta_1 p^2$, p prime and $\Delta_1 < -4$ this is an immediate corollary [4, Theorem 7.24, page 146]. \square

Thus we are able to control the order of the kernel and consequently set up a Schnorr analogue using the group $\text{Ker}(\phi_{Cl}^{-1})$ instead of \mathbb{F}_p^* .

3 The New Signature Scheme

In this section we will show how one can set up a NICE-Schnorr-type signature scheme using $\text{Ker}(\phi_{Cl}^{-1})$ instead of \mathbb{F}_p^* .

The *system setup* for Alice consists of the following steps:

1. Choose a random prime r and set $\Delta_1 = -r$ if $r \equiv 3 \pmod{4}$ or $\Delta = -4r$ otherwise.
2. Choose a random prime q , which will later on serve as the order of the used subgroup of $\text{Ker}(\phi_{Cl}^{-1}) \subset Cl(\Delta_p)$.
3. Choose a random prime p , such that $\left(\frac{\Delta_1}{p}\right) = 1, q|(p-1)$ and set $\Delta_p = \Delta_1 p^2$.
4. Choose a random $\alpha = x + y\omega$ such that $\varphi(\alpha\mathcal{O}_{\Delta_1})$ is of order q in $Cl(\Delta_p)$. This may be done by choosing a random $\beta = x' + y'\omega$ and computing $\alpha = \beta^{(p-1)/q}$ until $\mathfrak{g} = \rho_p(\varphi(\alpha\mathcal{O}_{\Delta_1})) \neq \mathcal{O}_{\Delta_1}$ using the algorithm Gen-Exp from [9] or the more efficient CRT variant introduced in Section 5.
5. Choose a random integer $a < q$ and compute the public key $\mathfrak{a} = \rho_p(\mathfrak{g}^a)$.
6. The secret key of Alice is the triple x, y, a .

Note that Alice will keep secret p, q, r, x, y, a and only publishes $\Delta_p, \mathfrak{g}, \mathfrak{a}$. Now the signature generation and verification procedure is analogous to the original Schnorr-scheme [20]. The only difference is that Alice may speed up the signature generation process by using the knowledge of $\alpha = x + y\omega$ and performing the computation in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ instead of using the fairly inefficient ideal arithmetic.

More precisely Alice performs the following steps to sign a message $m \in \mathbb{Z}$:

1. Choose a random integer $1 < k < q$ and compute $\mathfrak{k} = \text{Gen-CRT}(x, y, p, k)$, where the algorithm Gen-CRT() is given in Section 5.
2. Compute $e = h(m||\mathfrak{k})$ and $s \equiv ae + k \pmod{q}$.
3. Alice's signature for m is the pair (e, s) .

The verification is completely analogous to the original scheme [20] using standard ideal arithmetic (see e.g. [3]) in the *non-maximal* order:

1. Compute $\mathfrak{v} = \rho_p(\mathfrak{g}^s \mathfrak{a}^{-e})$ and $e' = h(m||\mathfrak{v})$.
2. The signature is valid if and only if $e' = e$.

It is clear that the verification works if the signature was generated by Alice, because $\mathfrak{v} \sim \mathfrak{g}^s \mathfrak{a}^{-e} \sim \mathfrak{g}^s \mathfrak{g}^{-ae} \sim \mathfrak{g}^k \sim \mathfrak{k}$. Thus $h(m||\mathfrak{k}) = h(m||\mathfrak{v})$ and hence $e' = e$.

While the procedures for signature generation and verification are completely analogous to the original scheme there is a big difference in the overall scheme which has to be considered more closely. Our scheme is (beside other difficulties which are explained below) based on the intractability of factoring Δ_p . In step 2. the s -part of the signature is unique modulo q , which is kept secret. Thus by collecting a lot of signatures with different s 's one may hope to learn the magnitude of q , which divides $p-1$. This information *might* be useful to factor Δ_p . In the next section however we will show that such an attack is no real threat.

4 Security Issues of the Proposed Scheme

In this section we will discuss the security and appropriate parameter sizes of our proposed scheme. As it relies on the difficulty of computing discrete logarithms in (a subgroup of) the kernel of ϕ_{Cl}^{-1} we will start with relating this problem to more conventional problems such as factoring and computing logarithms in finite fields.

The following result shows that "in practice" the DL-problem in $\text{Ker}(\phi_{Cl}^{-1})$ is "about as hard" as factoring Δ_p .

Theorem 1. *With notations as in the previous section we have the following two probabilistic polynomial time reductions:*

1. *Factoring the discriminant Δ_p can be reduced to the DL-problem in $\text{Ker}(\phi_{Cl}^{-1})$.*
2. *If the factorization of $\Delta_p = \Delta_1 p^2$ is known, then one can reduce the DL-problem in $\text{Ker}(\phi_{Cl}^{-1})$ to the DL-problem in \mathbb{F}_p^* .*

Proof:(Sketch) To show 1. we assume that some oracle is able to compute discrete logarithms in $\text{Ker}(\phi_{Cl}^{-1})$. That is given a fixed generator \mathbf{g} of $\text{Ker}(\phi_{Cl}^{-1})$ it returns on input of some element $\mathbf{g}^e \in \text{Ker}(\phi_{Cl}^{-1})$ the smallest possible exponent e . It is easy to see that this oracle can be used to determine $|\text{Ker}(\phi_{Cl}^{-1})| = p \pm 1$, where the '+' occurs for our concrete setup. This is done by choosing some e' and handing over $\mathbf{g}^{e'}$ to the oracle. Then $e' > |\text{Ker}(\phi_{Cl}^{-1})|$ implies that $e' > e$ and then $e' - e = k|\text{Ker}(\phi_{Cl}^{-1})|$ for some integer k . If we repeat this step multiple times then the gcd of the obtained differences will be $|\text{Ker}(\phi_{Cl}^{-1})|$ with high probability. The reduction 2. is shown in [10]. \square

Note that as in the original Schnorr-setup our scheme operates in a *subgroup* of the kernel. While it is not rigorously proven it is commonly assumed that the DL-problem in the subgroup is indeed as hard as the "full" DL-problem. Thus the assumption that the DL-problem in a subgroup of $\text{Ker}(\phi_{Cl}^{-1})$ is computational equivalent to the DL-problem in $\text{Ker}(\phi_{Cl}^{-1})$ itself is denoted by (subgroup-DL).

If we furthermore assume that our hash function acts like a random oracle [1], denoted by (ROM), then it is easy to prove the following result in complete analogy to [19, Theorem 5]:

Theorem 2. *Assume (ROM). If an existential forgery of the NICE-Schnorr signature scheme, under an adaptively chosen message attack, has non-negligible probability of success, then the discrete logarithm in subgroups of $\text{Ker}(\phi_{Cl}^{-1})$ can be solved in polynomial time.*

Thus we may combine the above results to obtain the following:

Corollary 2. *Assume (ROM) and (subgroup-DL). Furthermore assume that one is able to compute discrete logarithms in \mathbb{F}_p^* , which is feasible for the proposed parameter sizes. Then forging of signatures in our NICE-Schnorr-scheme is equivalent to factoring $\Delta_p = \Delta_1 p^2$.*

Thus our scheme is secure as long as we choose the parameters as follows:

- That one cannot compute discrete logarithms in $Cl(\Delta_p) \supset \text{Ker}(\phi_{Cl}^{-1})$ using a subexponential algorithm [12] we require $\Delta_p > 2^{400}$.
- That one cannot use a generic technique to compute discrete logarithms in the subgroup of the kernel, such as Pollard's ρ -method, we require $q > 2^{160}$.
- If one is able to factor Δ_p then one can reduce the discrete logarithm problem in the kernel to the discrete logarithm problem in \mathbb{F}_p^* using the recent reduction from [10]. Thus we need to ensure that Δ_p cannot be factored. With current algorithms (NFS and optimized ECM [18]) this should be impossible if we require $\Delta_p > 2^{720}$ and $p, r > 2^{240}$.
- As we do not disclose q , where $q|(p-1)$, we need to take care that the knowledge of many signatures does not help to find q , and hence breaking the scheme by factoring Δ_p , easier.

In the following we will discuss the potential threat of estimating q with the knowledge of many signatures more thoroughly. We will only sketch the main ideas here.

Estimating q by the maximal value of s

It is clear that $q|(p-1)$ and our scheme can be easily broken if $\Delta_p = \Delta_1 p^2$ is factored, because in this case one uses the result of [10] and is just faced with the computation of discrete logarithms in \mathbb{F}_p^* , which is possible for the proposed size of p . While it is not clear at the moment whether knowledge of q will immediately imply the knowledge of p , we will show that collecting signatures and taking care of the maximum s -value does not even help to come very close to q .

Since for all signatures (e_i, s_i) it holds that $s_i \in [0, q-1]$ an attacker could try to estimate q by the $\max_i(s_i)$ and then to use this gained information to factor Δ_p . The following argument shows that with overwhelming probability $\max_i(s_i)$ is not close enough to q .

Since h is a strong hash function we may assume that $e = h(m||\mathbf{k})$ and k are not significantly correlated for randomly chosen m and k . For simplicity we assume that e and k are statistically independent. Then $s \equiv ae + k \bmod q$ is uniform distributed in $[0, q-1]$. Therefore, for fixed $0 \leq \alpha \leq 1$ and for randomly chosen messages m_1, \dots, m_n and randomly chosen k_1, \dots, k_n with probability α^n the inequality $s_i \leq \alpha(q-1)$ holds for all $i \leq n$. Thus for $n \ll m$ the probability that $s_i \leq (1 - 1/m)q$ holds for all $i \leq n$ is approximately $(1 - n/m)$.

If we assume that the number n of signatures is limited by 2^{20} we can estimate that for all, say $\ell > 20$ at most with probability $2^{20-\ell}$ there is an $i \leq n$ with $s_i > (1 - 2^{-\ell})q$. On the other hand if $q \approx 2^{160}$ an attacker using the estimation

$$\max_i(s_i) \leq \alpha q \tag{4}$$

with $\alpha < (1 - 2^{-\ell})$ still has a search space of $2^{-\ell} \max_i(s_i) \approx 2^{160-\ell}$ many possible values q satisfying (4). Now if we assume that the time needed for finding q is about the square root of the size of the search space (i.e. $2^{80-\ell/2}$) we can estimate

the expected workload of this attack by $2^{80-\ell/2}/2^{20-\ell} = 2^{60+\ell/2} > 2^{70}$ which would be a formidable task. Note that to our knowledge there is no way to use that q is prime if one applies a "square root" algorithm such as Pollard- ρ to determine q .

Security problem of Schnorr-analogue in prime order subgroups of $(\mathbb{Z}/n\mathbb{Z})^*$ - Immunity of our scheme

The main practical advantage of our proposed scheme compared to the original one is its very efficient signature generation due to application of CRT for $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$. Thus one may think about a Schnorr-analogue operating in a prime order subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$ rather than \mathbb{F}_p^* , which would allow the same speedup by using CRT for $(\mathbb{Z}/n\mathbb{Z})^*$ in this case.

However we will briefly show in the following that, in contrary to our scheme, the breaking of such an analogue in $(\mathbb{Z}/n\mathbb{Z})^*$, is surprisingly easy.

Let $n = p_1 p_2$, p_1, p_2 prime and g be of order q in $(\mathbb{Z}/n\mathbb{Z})^*$, where q is prime. Because, $g^q \equiv 1 \pmod{n}$ we know by the CRT that $g^q \equiv 1 \pmod{p_1}$ and $g^q \equiv 1 \pmod{p_2}$. Thus it is clear that q must divide at least one of the numbers $p_1 - 1$ or $p_2 - 1$. W.l.o.g. we may assume that

$$q|(p_1 - 1) \tag{5}$$

Now there are two different cases to consider:

1. $q \nmid (p_2 - 1)$:

Because g is of order q in $(\mathbb{Z}/n\mathbb{Z})^*$ (and by (5) also in $\mathbb{F}_{p_1}^*$), we have $g \not\equiv 1 \pmod{p_1}$.

But $g^q \equiv 1 \pmod{p_2}$ together with $q \nmid (p_2 - 1)$ and the primeness of q implies that $g \equiv 1 \pmod{p_2}$ and hence $g - 1 = p_2 k$ for some integer k . Thus n is easily factored by computing $p_2 = \gcd(g - 1, n)$. Note that in this case one does not even need to know q and this case is very likely if p_2 is chosen randomly.

2. $q|(p_2 - 1)$:

In this case the scheme is similar to [5] and as shown in [14] is not immediately broken, but factoring n is made much easier, if one knows q .

We have $n = (2qp'_1 + 1)(2qp'_2 + 1)$, for some (in the worst case prime) numbers p'_1, p'_2 .

Then by [14, Proposition 2] one can factor n in $O(\frac{p'_1 + p'_2}{q})$ steps. Thus if p'_1, p'_2, q are the same order of magnitude this is a trivial task. Hence one must take great care, when working with subgroups of $(\mathbb{Z}/n\mathbb{Z})^*$.

The situation for our proposed scheme in $\text{Ker}(\phi_{Cl}^{-1}) \subset Cl(\Delta_p)$ is *entirely different*, because there is *no gcd-analogue* known for imaginary quadratic class groups, which could be applied to mount an "attack" like explained in the first situation above. Note that the existence of such a gcd-analogue would also imply the insecurity of NICE. The situation that $q|h(\Delta_1)$, which corresponds to the second situation above, is *very unlikely* if q, Δ_1 are chosen at random.

5 More Efficient Exponentiation Using CRT in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ and Timings

In Section 2 and [9] we saw that the arithmetic in $\text{Ker}(\phi_{Cl}^{-1})$ can be reduced to arithmetic in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$, which is much more efficient. In this section we will introduce a method which again speeds up the signing process considerably.

We will start with an auxilliary result.

Lemma 1. *Let \mathcal{O}_{Δ_1} be the maximal order and p be prime. Then there is an isomorphism between rings*

$$(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1}) \simeq \mathbb{F}_p[X]/(f(X)),$$

where $(f(X))$ is the ideal generated by $f(X) \in \mathbb{F}_p[X]$ and

$$f(X) = \begin{cases} X^2 - \frac{\Delta_1}{4}, & \text{if } \Delta_1 \equiv 0 \pmod{4}, \\ X^2 - X + \frac{1-\Delta_1}{4}, & \text{if } \Delta_1 \equiv 1 \pmod{4}. \end{cases} \quad (6)$$

Proof: See [10, Proposition 5]. □

This isomorphism between rings clearly implies an isomorphism between the multiplicative groups and with a little more effort we can show the central result of this section.

Theorem 3. *Assume that $\left(\frac{\Delta_1}{p}\right) = 1$ and the roots $\rho, \bar{\rho} \in \overline{\mathbb{F}}_p$ of $f(X) \in \mathbb{F}_p[X]$ as given in (6) are known. Then the following isomorphism can be computed in time $O((\log p)^2)$:*

$$(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \simeq \mathbb{F}_p^* \otimes \mathbb{F}_p^*$$

Proof: From Lemma 1 we know that there is an isomorphic map $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \rightarrow \mathbb{F}_p[X]/(f(X))^*$, where $f(X) \in \mathbb{F}_p[X]$ is given in (6). And that this isomorphism is trivial to compute.

Because $\left(\frac{\Delta_1}{p}\right) = 1$ the polynomial $f(X)$ is not irreducible, but can be decomposed as $f(X) = (X - \rho)(X - \bar{\rho}) \in \mathbb{F}_p[X]$ where $\rho, \bar{\rho} \in \mathbb{F}_p$ are the roots of $f(X)$. Thus if $\Delta_1 \equiv 0 \pmod{4}$ and $D = \Delta_1/4$ we have $\rho \in \mathbb{F}_p$ such that $\rho^2 \equiv D \pmod{p}$ and $\bar{\rho} = -\rho$. In the other case $\Delta_1 \equiv 1 \pmod{4}$ we have $\rho = (1+b)/2$, where $b^2 \equiv \Delta_1 \pmod{p}$ and $\bar{\rho} = (1-b)/2 \in \mathbb{F}_p$. Thus we have the isomorphisms

$$(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \simeq \left(\mathbb{F}_p[X]/(X - \rho) \right)^* \otimes \left(\mathbb{F}_p[X]/(X - \bar{\rho}) \right)^* \simeq \mathbb{F}_p^* \otimes \mathbb{F}_p^*.$$

Let $\alpha = a + b\omega \in (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ then the mapping $\psi : (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \rightarrow \mathbb{F}_p^* \otimes \mathbb{F}_p^*$ is given as $x_1 = \psi_1(\alpha) = a + b\rho \in \mathbb{F}_p^*$ and $x_2 = \psi_2(\alpha) = a + b\bar{\rho} \in \mathbb{F}_p^*$. The inverse map ψ^{-1} is computed by solving the small system of linear equations.

I.e. one will recover $a, b \in \mathbb{F}_p^*$ by computing $b = \frac{x_2 - x_1}{\bar{\rho} - \rho}$ and $a = x_1 - b\rho$. Thus both transformations ψ and ψ^{-1} need time $O((\log p)^2)$. \square

With this result we immediately obtain the of the following algorithm.

Algorithm 4 (*Gen-CRT*)

Input: $\alpha = x + y\omega \in \mathcal{O}_{\Delta_1}$, the conductor p , such that $\gcd(\mathcal{N}(\alpha), p) = 1$, $\left(\frac{\Delta_1}{p}\right) = 1$, the roots $\rho, \bar{\rho} \in \mathbb{F}_p^*$ of $f(X)$ as given in (6) and the exponent $n \in \mathbb{Z}$.

Output: $\mathfrak{a} = (a, b) = \rho_p(\varphi((\alpha\mathcal{O}_{\Delta_1})^n))$.

1. *IF* $n = 0$ *THEN* *OUTPUT*(1, $\Delta_1 \pmod{2}$)
2. *IF* $n < 0$ *THEN* $n \leftarrow -n$, $y \leftarrow -y$
3. $x_1 \leftarrow (x + py)^n \pmod{p}$
4. $x_2 \leftarrow (x + \bar{\rho}y)^n \pmod{p}$
5. $r \leftarrow (\bar{\rho} - \rho)^{-1} \pmod{p}$
6. $y_h \leftarrow (x_2 - x_1)r \pmod{p}$
7. $x_h \leftarrow x_1 - y_h\rho \pmod{p}$
8. /* Compute the standard representation $\mathfrak{A} = d(a, b) = \alpha_h \mathcal{O}_{\Delta_1}$ */
 - 8.1 /* Use $\frac{x+y\sqrt{\Delta_1}}{2}$ -form */
 - $x_h \leftarrow 2x_h$
 - IF* $\Delta_1 \equiv 1 \pmod{4}$ *THEN* $x_h \leftarrow x_h + y_h$
 - 8.2 Compute $d \leftarrow \gcd(y_h, (x_h + y_h\Delta_1)/2) = \lambda y_h + \mu(x_h + y_h\Delta_1)/2$, for $\lambda, \mu \in \mathbb{Z}$
 - 8.3 $A \leftarrow |x_h^2 - \Delta_1 y_h^2|/(4d^2)$
 - 8.4 $B \leftarrow (\lambda x_h + \mu(x_h + y_h)\Delta_1/2)/d \pmod{2A}$
9. /* Lift $\mathfrak{A}' = (1/d)\mathfrak{A}$ to the non-maximal order and reduce it */
 - $b \leftarrow Bf \pmod{2A}$
 - $(a, b) \leftarrow \rho_p(A, b)$
10. *OUTPUT*(a, b)

Correctness: The correctness of the exponentiation part is immediate, because we just compute the isomorphism ψ as given in the proof of Theorem 3, perform two exponentiations in \mathbb{F}_p^* and compute ψ^{-1} . The rest of the algorithm is equal to this part in Gen-Exp [9, Algorithm 19]. \square

Note that the computation of r in Step 5 can be done in a precomputation phase, as is it independent of the current α .

Finally we will give the timings of a first implementation using the LiDIA - package [13]. The timings of the exponentiation are given in microseconds on a Pentium 133 MHz. We used a random exponent $k < 2^{160}$ and $\Delta_p = \Delta_1 p^2$ where $\Delta_1 = -q$ (or $\Delta_1 = -4q$ if $q \equiv 1 \pmod{4}$ respectively) and p, q with equal bitlength. This may be compared with the timings for an exponentiation in $\mathbb{F}_{p'}^*$, where p' has the same bitlength as Δ_p . We neglected the time for hashing and computing the s -value.

One should note that the implementation of neither variant is optimized. This is no problem, because we are interested in the comparison, rather than the absolute timings.

group	\mathbb{F}_p^*	$\text{Ker}(\phi_{CI}^{-1})$	
arithmetic	modular	Gen-exp [9]	Gen-CRT
bitlength of	p	Δ_p	Δ_p
600	188	159	83
800	302	234	123
1000	447	340	183
1200	644	465	249
1600	1063	748	409
2000	1454	1018	563

Table 1. Timings for Schnorr-signature generation

These timings show that the signature generation of our proposed scheme using the novel CRT variant for exponentiation is *more than twice as fast as the original scheme*. While the signature verification is much less efficient than in the original scheme, this should be no problem, as the verification is usually not performed in a device with limited computational power, such as a smartcard.

6 Conclusion and Future Work

We have introduced a new signature scheme based on non-maximal imaginary quadratic orders, which features very fast signature generation. The security analysis shows that using standard assumptions the forging of signatures is equivalent to factoring $\Delta_p = \Delta_1 p^2$. Thus beside further studying implementation issues of cryptosystems based on non-maximal imaginary quadratic orders it will be an important task for the future to consider the factorization problem for this type of non-fundamental discriminant more closely.

References

1. M. Bellare, P. Rogaway: *Random Oracles are Practical: a Paradigm for Designing Efficient Protocols* in Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, ACM press, 1993, pp. 62-73 [21](#)
2. Z.I. Borevich and I.R. Shafarevich: *Number Theory* Academic Press: New York, 1966 [16](#)
3. H. Cohen: *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics **138**. Springer: Berlin, 1993. [16, 17, 20](#)
4. D.A. Cox: *Primes of the form $x^2 + ny^2$* , John Wiley & Sons, New York, 1989 [16, 18, 19](#)
5. M. Girault: *An identity based identification scheme based on discrete logarithms modulo a composite number*, Advances in Cryptology - Proceedings of Eurocrypt '90, LNCS 473, Springer, pp. 481-486 [23](#)

6. M. Hartmann, S. Paulus and T. Takagi: *NICE - New Ideal Coset Encryption*, to appear in proceedings of CHES, LNCS, Springer, 1999 [14](#)
7. D. Hühnlein, M.J. Jacobson, S. Paulus and T. Takagi: *A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption*, Advances in Cryptology - EUROCRYPT '98, LNCS **1403**, Springer, 1998, pp. 294-307 [14](#), [15](#), [16](#), [17](#), [18](#)
8. D. Hühnlein, A. Meyer and T. Takagi: *Rabin and RSA analogues based on non-maximal imaginary quadratic orders*, Proceedings of ICICS '98, ISBN 89-85305-14-X, 1998, pp. 221-240 [14](#), [15](#)
9. D. Hühnlein: *Efficient implementation of cryptosystems based on non-maximal imaginary quadratic orders*, to appear in proceedings of SAC'99, LNCS, Springer, 1999, preprint via <http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html> [14](#), [15](#), [16](#), [19](#), [20](#), [24](#), [25](#), [26](#)
10. D. Hühnlein, T. Takagi: *Reducing logarithms in totally non-maximal imaginary quadratic orders to logarithms in finite fields*, to appear in proceedings of ASIACRYPT'99, Springer, LNCS, 1999, preprint via <http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html> [14](#), [15](#), [16](#), [21](#), [22](#), [24](#)
11. D. Hühnlein: *A survey of cryptosystems based on imaginary quadratic orders*, forthcoming, 1999 [15](#)
12. M.J. Jacobson Jr.: *Subexponential Class Group Computation in Quadratic Orders*, PhD thesis, TU Darmstadt, to appear, 1999 [22](#)
13. LiDIA: *A C++ library for algorithmic number theory*, via <http://www.informatik.tu-darmstadt.de/TI/LiDIA> [25](#)
14. W. Mao: *Cryptoanalysis in Prime Order Subgroups of \mathbb{Z}_n^** , contribution to IEEE-P1363, manuscript via <http://www.ieee.org>, 1998 [16](#), [23](#)
15. National Institute of Standards and Technology (NIST): *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication 186, **FIPS-186**, 19th May, 1994
16. J. Neukirch: *Algebraische Zahlentheorie*, Springer, Berlin, 1992 [18](#)
17. S. Paulus and T. Takagi: *A completely new public key cryptosystem with quadratic decryption time*, to appear in Journal of Cryptology, 1998, preprint via <http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/sachar.html> [14](#), [15](#)
18. R. Peralta and E. Okamoto: *Faster factoring of integers of a special form*, IEICE Trans. Fundamentals, Vol. E-79-A, No. 4, 1996, pp. 489-493 [16](#), [22](#)
19. D. Pointcheval, J. Stern: *Security Proofs for Signature Schemes*, Proceedings of Eurocrypt '96, Springer-Verlag, LNCS **1070**, 1996, pp. 387-398 [21](#)
20. C.P. Schnorr: *Efficient identification and signatures for smart cards*, Advances in Cryptology - CRYPTO '89, LNCS **435**, 1990, pp. 239-252 [16](#), [20](#)

Identification of Bad Signatures in Batches

Jarosław Pastuszak¹, Dariusz Michałek¹, Josef Pieprzyk², and Jennifer Seberry²

¹ Systems Research Institute
Polish Academy of Sciences
Warsaw, POLAND

jarek.pastuszak@bsb.com.pl

² Centre for Computer Security Research
School of IT and Computer Science
University of Wollongong
Wollongong, NSW 2522, AUSTRALIA
Josef.Pieprzyk@uow.edu.au
Jennifer.Seberry@uow.edu.au

Abstract. The paper addresses the problem of bad signature identification in batch verification of digital signatures. The number of generic tests necessary to identify all bad signatures in a batch instance, is used to measure the efficiency of verifiers. The divide-and-conquer verifier $DCV_\alpha(x, n)$ is defined. The verifier identifies all bad signatures in a batch instance x of the length n by repeatedly splitting the input into α sub-instances. Its properties are investigated. In particular, probability distributions for the number of generic tests necessary to identify one, two and three bad signatures, are derived. The average numbers of GT tests necessary to identify bad signatures ranging from 1 to 16 are obtained from computer simulation. Further, a Hamming verifier (HV) is defined which allows to identify a single bad signature in a batch of the length $n = 2^k - 1$ using $k + 2$ tests. HV is generalised into the two-layer Hamming verifier (2HV). Given a batch instance of the length $2^k - 2$, the 2HV verifier identifies a single bad signature using $k + 2$ tests and two bad signatures in expense of $3k + 3$ tests. The work is concluded by comments about a general model for verification codes identifying t bad signatures and the design of verifiers using combinatorial structures.

1 Introduction

Digital signatures are main cryptographic tools for message authentication. Unlike hand-written signatures, digital ones differ from one document to another as they produce a fingerprint which reflects both the identity of signer (or more precisely their secret signing key) and the contents of the document (typically embedded in its digest). Any digital signature includes signing and verification algorithms. The signing algorithm can be run by the holder of the secret signing key. The verification algorithm can be run by everybody as the matching (verification) key is public.

Often a signature is generated once but its verification is done many times. A growing usage of digital signatures for electronic payment systems stresses the need for streamlining of the signature verification. Batch verification offers an efficient verification of a collection of related signatures at a cost of making a mistake. The probability of mistake can be traded off with the efficiency. Batch verification is an option if the signature used exhibits the homomorphic property.

The idea of batch verification was spelt out in many papers [2,4,6,8].

2 Motivation

Undoubtedly, fast signature verification seems to be of utmost importance when there is a need for continual processing of many signatures. As shown by Bellare, Garay and Rabin in [1] there are three generic test which can be used for fast batch verification of signatures. Efficiency of these tests varies and depends on the size of a signature batch being verified. The main problem with batch verification is that they trade efficiency with security. In the case of individual signature verification, an attacker is forced to break the underlying signature scheme if they want to generate a valid signature for a message. In the case when the batch verification is applied, the attacker may also explore weaknesses existing in the verification tests. Verification tests are probabilistic algorithms for which it is possible to set the bound on the probability of acceptance of invalid signatures in the batch tested. As there is a direct relation between the probability and efficiency, one can expect that the probability may be lowered during the time when the heavy processing is expected (typically, the end of the week). Instead of breaking the underlying signature, attackers are encouraged to generate messages with invalid signatures on a massive scale. This serves two purposes. The first purpose is to increase the verification load, and one can expect that the manager responsible for verification of signatures, will lower the threshold probability even further. The second purpose is to increase the probability of attacker success. On the top of this, the attacker may have specific knowledge about which test will be used and what parameters are employed. This knowledge may give some hints as to how invalid signatures could be produced to maximise the chance of slipping through the tests.

When a collection of signatures passes the tests, the verifier accepts all the signatures as valid. Otherwise, the collection is rejected. Now the verifier must separate the valid signatures from invalid ones. In this paper, we consider different methods of invalid signature identification and evaluate efficiency of tests with invalid signature identification.

3 Background

There are two homomorphic operations widely used for signing: modular exponentiation (the base is fixed) and RSA exponentiation (the exponent is fixed). Consider modular exponentiation defined for a cyclic group of order q , where g is the cyclic group generator. The DSA or DSS signatures and their versions are

signatures of this kind. Being more precise, the exponents are computed individually for each signature. This computation is cheap – it takes one modular inversion and multiplication. The final verification can be done in batches in which exponents are added (see [5]).

Given a batch $x = ((m_1, s_1), \dots, (m_n, s_n))$ of messages with their signatures, signatures can be verified one by one by checking

$$g^{m_i} \stackrel{?}{=} s_i \text{ for } i = 1, \dots, n$$

The cost of verification is n exponentiations. To reduce the number of expensive exponentiations and speed up the verification process, one can verify the following

$$V_g(x) \equiv \left(g^{\sum_{i=1}^n m_i} \stackrel{?}{=} \prod_{i=1}^n s_i \right) \quad (1)$$

This costs one exponentiation, $n - 1$ modular multiplications, and $n - 1$ modular additions. Typically, the calculation of $\sum_{i=1}^n m_i$ is done modulo q while $\prod_{i=1}^n s_i$ is performed modulo p where q divides $p - 1$.

Consider the RSA exponentiation where the modulus N is the product of two primes p and q . The signer secret key is d and the public verification key is e . All signed messages are smaller than the modulus N . A typical batch of signatures looks like $((m_1, s_1), \dots, (m_n, s_n))$. Sequential verification of the batch

$$s_i^e \stackrel{?}{=} m_i \text{ for } i = 1, \dots, n,$$

takes n exponentiations. Again, the verification process can be sped up by using

$$V_e(x) \equiv \left(\prod_{i=1}^n m_i \stackrel{?}{=} \left(\prod_{i=1}^n s_i \right)^e \right) \quad (2)$$

This takes one exponentiation and $2(n - 1)$ modular multiplications.

In general, a batch verifier is a probabilistic algorithm B which takes a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ and a security parameter ℓ . The algorithm

- outputs “0” always whenever all the signatures in the batch are correct,
- outputs “1” with probability $1 - 2^{-\ell}$ whenever the batch contains incorrect signatures.

A batch verifier never makes mistakes when the batch is “clean”. If the batch is “dirty” or contains incorrect signatures, then the batch verifier makes mistakes with probability $2^{-\ell}$.

There is a universal test which is applicable for any signature scheme which has a homomorphic property. The test (in [1] called random subset test) is defined as follows.

Definition 1. *Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ and a security parameter ℓ . The universal test (UT) takes ℓ rounds. For each round*

1. pick a random set $\mathcal{T} = \{t_1, \dots, t_n\}$, i.e. each t_i is selected independently and with the same probability from $\{0, 1\}$,
2. create a subset $x_{\mathcal{T}} = \{(m_i, s_i) | t_i = 1\}$,
3. run the test $V(x_{\mathcal{T}})$ (either $V_g(x_{\mathcal{T}})$ or $V_e(x_{\mathcal{T}})$). If the test accepts go to the next round. Otherwise, reject the batch.

A useful test for signatures based on a fixed base applies a random string of small integers used in the test as exponents (in [8] called small exponents test).

Definition 2. Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ and a security parameter ℓ . The small exponent (SE) test:

1. select at random a collection of small integers $e = (e_1, \dots, e_n)$ where $e_i < 2^\ell$,
2. convert the instance x into $x_e = ((m_1 e_1, s_1^{e_1}), \dots, (m_n e_n, s_n^{e_n}))$,
3. run the test $V_g(x')$. If the batch instance x' passes the test accept x otherwise reject.

Clearly, we are interested in a generic test which always succeed when all signatures are valid and fails with an overwhelming probability when there is one or more bad signatures.

Definition 3. Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$. The generic test (GT) takes a batch instance x and

1. outputs “0” whenever all signatures are valid. The test never makes mistakes for this case,
2. returns “1” whenever there is at least one bad signature. In this case the test makes mistakes with probability $2^{-\ell}$.

If a batch of signatures passes tests, then the verifier accepts the whole batch. The probability of mistake can be make small enough say smaller than 2^{-100} . However when a batch fails a test, the verifier is not able to reject all signatures in the batch. The verifier faces the problem of identification of bad signatures. Let us consider some possible solutions for bad signature identification.

The simplest solution for it could be based on testing all signatures one by one using the GT test.

Definition 4. Naive Verifier. Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$.

1. Run $GT(x, n)$. If $GT(x, n)=0$, accept the instance x and exit. Otherwise, when $GT(x, n)=1$, for $i = 1$ to $i = n$ do:
 - apply $GT(x_i, 1)$,
 - if $GT(x_i, 1) = 1$ then store x_i otherwise go for the next i .
2. Output all stored signatures in the list $NV(x)$.

where $x_i = (m_i, s_i)$.

The well-known twelve-coin problem is very much related to the identification of bad signatures. It can be formulated as follows.

Given 12 coins all of equal weight, except one defective coin. It is not known whether the defective coin is lighter or heavier than each of the others. Assume that there is a set of two-dish scales which can be used to carry out tests. Coins can be placed on both sides and if the weights are equal then the scales balance, otherwise they tilt downwards on the side carrying the heavier weight.

Find a sequence of tests which can be performed using the scales to identify the defective coin within the three weightings only.

Note that identification of a bad signature resembles the twelve-coin problem. The main difference is that tests performed on batches do not allow us to see how the scales tilt. In other words, the tests carried out on batches allow us to see whether the batch is clean (the scales balance) or dirty (the scales do not balance).

4 Divide-and-Conquer Verifiers

Identification of bad signatures can be implemented by the so called divide-and-conquer (DC) verifier. The idea seems to be straightforward and can be traced in the literature under the name “cut and choose” [4].

The verifier is an algorithm which takes a batch instance x and outputs either “0” when the batch instance is clean otherwise returns a list of all bad signatures. It is defined as a recursive function.

Definition 5. DC Verifier. Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ with $n = 2^k$ signatures.

1. Stopping case: If the instance consists of $n = 1$ signature, then run the generic test on the input, i.e. $GT(x, 1)$. If $GT(x, 1) = 0$, return 0 and exit. Otherwise output the bad signature and exit.
2. If the instance consists of $n \neq 1$ signature, apply the generic test on the input sample, i.e. $GT(x, n)$. If $GT(x, n) = 0$, return 0 and exit. Otherwise go to the recursive step.
3. Recursive step: Divide the instance x into α batch instances (x_1, \dots, x_α) containing $\frac{n}{\alpha}$ signatures each. The division is done at random. Call the DC verifier for α sub-instances, i.e. $DCV_\alpha(x_1, \frac{n}{\alpha}) \dots DCV_\alpha(x_\alpha, \frac{n}{\alpha})$.

The computational overhead of our verifiers is measured by the number of times the GT test is called during verification process. The worst case occurs when a batch instance contains all bad signatures. So the maximum number of tests performed by DCV_α is

$$\# \max(DCV_\alpha, n) = \sum_{i=0}^k \alpha^i = \frac{\alpha^{(k+1)} - 1}{\alpha - 1} = \frac{n\alpha - 1}{\alpha - 1} \quad (3)$$

where α indicates that the DCV verifier slices input instances into α sub-instances of the same length and n is the length of the input batch instance.

From Equation (3), it is easy to observe that for very badly contaminated instances, the selection of a large α is preferred. Note that if $\alpha = n = 2^k$, then the DCV verifier becomes the NV verifier which always consumes $n + 1$ tests.

4.1 Degree of Contamination Versus Parameter α

It is interesting to ask about the degree of contamination of batch instances for which the naive verifier becomes more efficient than DCV_2 . This is an important issue for efficient signature verification. To answer this question, assume that a batch instance consists of $n = 2^k$ signatures contaminated with $t = 2^r$ bad signatures ($r < k$). Denote the maximum numbers of GT tests necessary to identify all t bad signatures out of total n ones using the NV and DCV_2 verifiers by $\# \max(NV, n, t)$ and $\# \max(DCV_2, n, t)$, respectively.

Note that the worst case occurs when the DC verifier after the r -th recursive step all sub-instances contain precisely one bad signature. To get to this point, DCV_2 consumes precisely $2^r - 1$ tests. So

$$\# \max(DCV_2, 2^k, t) = 2^r - 1 + 2^r \times \# \max(DCV_2, 2^{k-r}, 1).$$

A single bad signature in a batch instance of size 2^{k-r} is always identifiable using $2(k-r) + 1$ tests. Therefore, we obtain

$$\# \max(DCV_2, 2^k, t) = 2^{r+1}(k-r+1) - 1.$$

Now we can ask how small the contamination of a batch instance should be to render the DCV verifier more efficient or

$$\# \max(DCV_2, 2^k, t) < \# \max(NV, 2^k, t).$$

If we substitute values obtained, then the inequality becomes

$$2^{(r+1)}(k-r+1) - 1 < 2^k + 1$$

or equivalently

$$k - r + 1 < 2^{k-r-1} + 2^{-r}.$$

It is easy to check that this inequality holds for any $k - r \geq 3$. So we have proved the corollary.

Corollary 1. *DCV_2 is more efficient (consumes less GT tests) from the NV verifier if batch instances of 2^k signatures contain less than 2^{k-3} bad ones.*

Note that we have compared DCV_2 (binary split of batch instances) with DCV_n (equivalent to NV). Similar considerations can be made for any two verifiers DCV_α , DCV_β for $\alpha \neq \beta$. This makes sense if the contamination varies and the parameter α can be adjusted accordingly.

Results of computer simulation conducted to determine the relation between the degree of contamination and the parameter α are summarised in Table 1.

Table 1. Tradeoff between parameter α and the degree of batch contamination

Number n	Number of bad signatures t	Optimal Parameter α
128	1	2, 4
	2, 4	4, 8
	8	32
	16	32, 64
	32	128
256	1	2, 4
	2, 4, 8	4
	16	8
	32, 64	64
512	1	2, 4
	2, 4, 8, 16	4
	32, 64	128
1024	1, 2	2, 4
	4, 8	4
	16, 32	4, 8
	64	256
	128	512
2048	1	2, 4
	2, 4, 8, 16, 32, 64	4
	128, 256	512
4096	1	2, 4
	2, 4, 8, 16, 32, 64, 128	4
	256, 512	1024

4.2 Number of Tests Needed to Identify t Bad Signatures

Denote $\#(DCV_\alpha, n, t)$ to be the number of GT tests necessary to identify bad signatures from a batch instance with n signatures provided t ones are bad. As the DCV_α verifier is probabilistic in its nature, the number $\#(DCV_\alpha, n, t)$ is in fact a random variable. To simplify our notation, let

$$N_\alpha(t, n) = \#(DCV_\alpha, n, t).$$

Our aim is to derive the probability distribution for the variable $N_2(t, n)$.

Consider the verifier DCV_2 and the corresponding random variable $N_2(t, n)$. Let $t = 1$. Obviously, the verifier needs to perform $2k + 1$ tests, i.e.

$$N_2(1, 2^k) = 2k + 1.$$

This number of tests is constant and occurs with probability 1. By the way, the number of tests can be cut almost by half if $t = 1$ is known before hand as $N_2(1, 2^k) = k + 1$. This observation of course may be used for optimisation of

the DCV verifier. This is especially effective for $\alpha = 2$. If a sub-instance passes the GT test, the second sub-instance is not tested (as it must fail it anyway). Instead, it is divided into halves and one of the resulting sub-instances is tested.

Let $t = 2$. Note that random variable $N_2(2, 2^k)$ can be expressed by random variables $N_2(2, 2^{k-1})$ and $N_2(1, 2^{k-1})$ according to the following equation:

$$N_2(2, 2^k) = \begin{cases} 1 + 2N_2(1, 2^{k-1}) & \text{with probability } p_{1,0} \\ 2 + N_2(2, 2^{k-1}) & \text{with probability } p_{2,0} \end{cases} \quad (4)$$

Similarly, we can write

$$N_2(2, 2^{k-1}) = \begin{cases} 1 + 2N_2(1, 2^{k-2}) & \text{with probability } p_{1,1} \\ 2 + N_2(2, 2^{k-2}) & \text{with probability } p_{2,1} \end{cases} \quad (5)$$

For $i = 2, \dots, k - 1$, we can generalise as

$$N_2(2, 2^{k-i}) = \begin{cases} 1 + 2N_2(1, 2^{k-i-1}) & \text{with probability } p_{1,i} \\ 2 + N_2(2, 2^{k-i-1}) & \text{with probability } p_{2,i} \end{cases} \quad (6)$$

Assume that at step j , two bad signatures clustered together in a single instance have been put into two different sub-instances. This means that the bad signatures were placed in the same instance j times in a row. Therefore

$$N_2(2, 2^k)(j) = 2j + 1 + 2N_2(1, 2^{k-j-1}) = 4k - 2j - 1 \quad (7)$$

where $j = 0, 1, \dots, k - 1$.

Now we are ready to calculate probabilities $p_{i,j}$. The parameter $n = 2^k$. The probability $p_{1,0}$ expresses the probability that the initial batch instance splits into two sub-instances containing one bad signature each so

$$p_{1,0} = \frac{\binom{2}{1} \binom{n-2}{\frac{n}{2}-1}}{\binom{n}{\frac{n}{2}}} = \frac{n}{2(n-1)}.$$

Similarly, the probability that after the split, one of the sub-instances contains two bad signatures is:

$$p_{2,0} = 2 \times \frac{\binom{2}{0} \binom{n-2}{\frac{n}{2}}}{\binom{n}{\frac{n}{2}}} = \frac{n-2}{2(n-1)}.$$

The multiplier 2 indicates the fact that two bad signatures can be in the first or the second sub-instance. Continuing our calculations, we obtain

$$p_{1,i} = \frac{n}{2(n-2^i)} \quad (8)$$

$$p_{2,i} = \frac{n-2^{i+1}}{2(n-2^i)} \quad (9)$$

The probability $p(j)$ that for some step j , two bad signatures have been placed into two different sub-instances is:

$$\begin{aligned} p(0) &= p_{1,0} \\ p(1) &= p_{2,0} \times p_{1,1} \\ &\vdots \\ p(j) &= p_{2,0} \times p_{2,1} \times \dots \times p_{2,j-1} \times p_{1,j} \end{aligned}$$

After substituting values, the above equation takes on the following form:

$$p(j) = \frac{n}{n-1} \frac{1}{2^{j+1}}$$

for $j = 0, 1, \dots, k-1$ and $n = 2^k$. So we have proved the following corollary.

Corollary 2. *Given the DCV verifier with $\alpha = 2$. If a batch instance of length $n = 2^k$ is contaminated by two bad signatures, then the number $N_2(2, n)$ of necessary GT tests is a random variable whose probability distribution is as follows:*

$$P(N_2(2, n) = 4k - 2j - 1) = \frac{n}{n-1} \frac{1}{2^{j+1}} \quad (10)$$

for $j = 0, 1, \dots, k-1$.

Now we derive the probability distribution for the required number of GT tests when the input batch instance is contaminated by three bad signatures ($t = 3$).

The number of GT tests is denoted by $N_2(3, 2^k)$. The number of tests satisfies the equation

$$N_2(3, 2^k) = \begin{cases} 1 + N_2(1, 2^{k-1}) + N_2(2, 2^{k-1}) & \text{with probability } p_1 \\ 2 + N_2(3, 2^{k-1}) & \text{with probability } p_2 \end{cases}$$

It means that after the first step, the verifier may split the input instance into two sub-instances where (1) one sub-instance contains one bad signature and the other sub-instance is contaminated by two bad signatures, (2) one sub-instance is clean and the other includes 3 bad signatures. The probability p_1 is equal to

$$p_1 = \frac{\binom{3}{1} \binom{n-3}{\frac{n}{2}-1}}{\binom{n}{\frac{n}{2}}} = \frac{3n}{4(n-1)}.$$

and the probability p_2 is

$$p_2 = 2 \times \frac{\binom{3}{0} \binom{n-3}{\frac{n}{2}}}{\binom{n}{\frac{n}{2}}} = \frac{n-4}{4(n-1)}.$$

Assuming that the bad signatures have been tossed into two sub-instances at the first step by the verifier, then the probability distribution can be derived from previous considerations (see Equation 8) and

$$P(N_2(3, n) = 6k - 2j - 5 | (1, 2)) = p_1 \times p_{1,1} = \frac{3n}{4(n-1)} \frac{n}{(n-2)} \frac{1}{2^{j+1}}$$

for $j = 0, 1, \dots, k-2$.

Consider the case when bad signatures have been tossed into the same sub-instance (the other sub-instance is clean) – the case (0,3). Assume that for certain step i , the three bad signatures have been split into either (1,2) or (2,1). It means also that three bad signatures were tossed together i times so

$$N_2(3, n) = 2i + 1 + N_2(1, 2^{k-i-1}) + N_2(2, 2^{k-i-1})$$

for $i = 0, \dots, k-1$. After substituting the expressions obtained for $t = 2$ and $t = 1$, we obtain final probability distribution.

Corollary 3. *Given the verifier DCV $_\alpha$ with $\alpha = 2$. If a batch instance of length $n = 2^k$ is contaminated by three bad signatures, then the number $N_2(3, n)$ of required GT tests is a random variable whose probability distribution is as follows:*

$$P(N_2(3, n) = 6k - 4i - 2j - 5) = \frac{3n^2}{(n-1)(n-2)} \frac{1}{2^{2i+j+3}} \frac{n - 2^{i+1}}{n - 2^{k-i-1}} \quad (11)$$

for $i = 0, 1, \dots, k-1$ and $j = 0, 1, \dots, k-i-2$.

Knowing the probability distributions for the number of GT tests necessary to identify bad signatures in the cases when $t = 1, 2, 3$, it is easy to find the average number of test. For the number of bad signatures $t > 3$, the average can be estimated using computer simulation. The results are compiled in Table 2.

4.3 Optimisation of DC Verifiers

As observed above, for DCV $_2$, the number of GT tests can be reduced if the verifier knows the precise number of bad signatures. If there is only a single bad signature ($t = 1$), then at each step the DCV $_2$ verifier needs to tests only single sub-instance out of two generated from the contaminated instance. If the sub-instance is clean, then the other sub-instance is dirty (and vice versa). So the number $N_2(1, 2^k) = 2k + 1$ can be reduced to $k + 1$. Even if the number of bad signatures is not known before hand, this observation can be exploited to reduce the number of GT tests.

Definition 6. Fast DC Verifier. *Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ with $n = 2^k$ signatures.*

1. Stopping case: *If the instance consists of $n = 1$ signature, then run the generic test on the input, i.e. $GT(x, 1)$. If $GT(x, 1) = 0$, exit. Otherwise output the bad signature and exit.*

Table 2. The average number of GT tests necessary to identify t bad signatures in a sequence of length n

t	n=16	n=32	n=64	n=128	n=256	n=512	n=1024
0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
1	9,0	11,0	13,0	15,0	17,0	19,0	21,0
2	13,5	17,3	21,2	25,1	29,1	33,0	37,0
3	17,1	22,5	28,2	34,0	39,8	45,8	51,7
4	19,9	26,9	34,3	41,9	49,7	57,6	65,5
5	22,2	30,8	39,8	49,3	58,9	68,7	78,5
6	24,1	34,2	44,9	56,0	67,5	79,2	91,0
7	25,6	37,2	49,5	62,4	75,7	89,3	103,0
8	27,0	39,9	53,8	68,4	83,5	99,0	114,7
9	28,0	42,4	57,8	74,1	91,0	108,3	125,9
10	28,9	44,6	61,5	79,5	98,2	117,4	136,9
11	29,6	46,6	65,1	84,7	105,1	126,1	147,6
12	30,2	48,5	68,4	89,6	111,8	134,7	158,0
13	30,6	50,2	71,5	94,3	118,3	143,0	168,2
14	30,9	51,7	74,5	98,9	124,5	151,1	178,2
15	31,0	53,1	77,3	103,3	130,6	159,0	188,0
16	31,0	54,4	80,0	107,5	136,6	166,7	197,6

2. If the instance consists of $n \neq 1$ signature, apply the generic test on the input sample, i.e. $GT(x, n)$. If $GT(x, n) = 0$, exit. Otherwise go to the recursive step.
3. Recursive step: Split the instance x into α batch instances (x_1, \dots, x_α) containing $\frac{n}{\alpha}$ signatures each. The split is done at random. Call the DC verifier for $\alpha-1$ sub-instances, i.e. $DCV(x_1, \frac{n}{\alpha}) \dots DCV(x_{\alpha-1}, \frac{n}{\alpha})$. If there is at least one dirty sub-instance, call $DCV(x_\alpha, \frac{n}{\alpha})$. Otherwise (i.e. if all sub-instances are clean), call the verifier $DCV(x_\alpha, \frac{n}{\alpha})$ in which the GT test is skipped.

Note that the fast verifier DCV_2 needs $\approx (1.5k + 1)$ tests (instead of $2k + 1$) if there is one bad signature (but the verifier does not know this before hand). The advantage drops if α grows. In general, the fast verifier DCV_α consumes $((\alpha - 1 + \frac{1}{\alpha})k + 1)$ tests instead of $(\alpha k + 1)$ assuming a single bad signature and the length of batch instance α^k .

Further improvement can be achieved if the split of instances is not random. It turns out that if the random split into sub-instances is replaced by deterministic split into α sub-instances, then the number $N_\alpha(t, n)$ preserve the same probability distribution assuming that the input batch instance is random. This assumption seems to hold in most practical situations.

Additionally, the DCV verifier can be sped up by a careful design of the GT test. To illustrate the point assume that the DCV verifier is used to identify bad signatures by running the test $V_e(x)$ defined by Equation (2). Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$. Note that the test $V_e(x)$ is run for the whole

instance x and needs to produce the product of all messages ($\prod_{i=1}^n m_i$) and all signatures ($\prod_{i=1}^n s_i$). Before calling the verifier, we can create two multiplication tables for messages and signatures. For instance, the message multiplication table is of the following form (the input instance is of length 16):

$$\begin{aligned} \text{Batch Instance: } & 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 \\ \text{1-st level of products: } & (1, 2)(3, 4)(5, 6)(7, 8)(9, 10)(11, 12)(13, 14)(15, 16) \\ \text{2-nd level of products: } & (1, 2, 3, 4)(5, 6, 7, 8)(9, 10, 11, 12)(13, 14, 15, 16) \\ \text{3-rd level of products: } & (1, 2, 3, 4, 5, 6, 7, 8)(9, 10, 11, 12, 13, 14, 15, 16) \\ \text{4-th level of products: } & (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16). \end{aligned}$$

where (i, j) stands for the product of $m_i \times m_j$. All multiplications needed by the DCV verifier are already stored in the tables. To run the test $V_e(x)$, it needs to perform a single exponentiation.

5 Verifiers Based on Hamming Codes

Assume that batch instances are contaminated by at most a single bad signature. This assumption is true most of the time when the source of errors is unreliable storage or communication so from time to time some signatures (or corresponding messages) get corrupted. Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ of length $n = 2^k - 1$ for some positive k . To identify a single bad signature, it is enough to design a Hamming code with the block length n and k parity check equations. Let H be a parity check matrix. H contains k rows and n columns. If the matrix H has the form

$$H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} = [1 \ 2 \ 3 \ \dots \ 2^k - 1]$$

where $h_i = (h_{i,1}, \dots, h_{i,n})$ is a binary string of length n with the weight 2^{k-1} and integers i in the matrix represent columns which are binary strings representing the integer. Note that the Hamming code with such H allows for a quick identification of error position as the error syndrome is the binary index of the position in which the error occurs (for details see [3]).

Definition 7. Hamming Verifier. *Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ of length $n = 2^k - 1$ for some positive k .*

1. *Apply the generic test on the input instance. If $GT(x, n)=0$, exit. Otherwise, go to the next step.*
2. *Create k sub-instances. i.e.*

$$x_i = \{(m_j, s_j) | h_{i,j} = 1\}$$

for $i = 1, \dots, k$ where x_i is a sub-instance composed from elements of x chosen whenever $h_{i,j}$ is equal to 1 (elements of x for which $h_{i,j}=0$ are ignored).

3. Run $GT(x_i, 2^{k-1}) = \sigma_i$ for $i = 1, \dots, k$ where $\sigma_i = 0$ if the test accepts x_i or $\sigma_i = 1$ if it fails. The syndrome $(\sigma_1, \dots, \sigma_k)$ identifies the position of the bad signature.
4. Apply the generic test on the input instance without the bad signature. If the batch instance is accepted, return the index of the bad signature. Otherwise, the verifier fails and exits 1.

The Hamming verifier (HV) succeeds whenever batch instances of the length $2^k - 1$ are contaminated by single bad signatures and HV consumes $k + 2$ GT tests. This number is almost identical to the number which is needed by DCV₂ when the verifier knows that there is a single bad signature in the batch.

Consider the case when HV fails – this obviously indicates that the number of bad signatures is greater than 1. There are at least two possible courses of action:

1. Filter out all clean signatures identified by HV. Consider the syndrome string $(\sigma_1, \dots, \sigma_k)$ generated by HV. Clearly, we can remove all clean sub-instances x_i for which $\sigma_i = 0$ and identify the bad signatures using DCV for the remainder of the batch.
2. Use the BCH code which corrects two errors to identify two bad signatures. This is an attractive option as we can reuse all results of GT tests obtained by HV. This gives rise to two level Hamming verifier defined below.

Unfortunately, BCH codes correcting two errors are not directly applicable. The main reason is different interactions of bad signatures compared to transmission errors in codes. Note that if two errors occur in a communication channel then they cancel each other in a parity check equation or more precisely, they obey the XOR addition. On the other hand, the behaviour of bad signatures is governed (with overwhelming probability) by logical addition. A parity checking equation failure does not depend on how many bad signatures it contains. This fact make the problem more difficult but also more interesting.

Assume that we have a batch of $n = 2^k - 2$ signatures which includes two bad ones ($t = 2$). As previously, we start from a Hamming code correcting a single error with the corresponding parity check matrix

$$H_1 = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} = [1 \ 2 \ \dots \ 2^k - 2] = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 1 \\ \vdots & \ddots & & \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Note that this matrix does not contain any column with all ones. We define another matrix

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \quad (12)$$

where H_1 is as defined above and H_2 is the negation of H_1 , i.e. $H_2(i, j) = \overline{H_1(i, j)}$ for $i, j = 1, \dots, k$.

Definition 8. Two-Layer Hamming Verifier. Given a batch instance $x = ((m_1, s_1), \dots, (m_n, s_n))$ of length $n = 2^k - 2$ for some positive k and a linear code represented by its parity check matrix H with $2k$ rows and n columns of the form given by Equation 12. Assume that the batch is contaminated by two bad signatures with their indices

$$I_1 = (i_{1,1}, \dots, i_{1,k}) \text{ and } I_2 = (i_{2,1}, \dots, i_{2,k})$$

1. Apply the generic test on the input instance. If $GT(x, n) = 0$, exit. Otherwise, go to the next step.
2. Create $2k$ sub-instances (or control groups) corresponding to rows of the matrix H or

$$\begin{aligned} x_{1,i} &= \{(m_j, s_j) | H_1(i, j) = 1 \text{ and } j = 1, \dots, n\} \\ x_{2,i} &= \{(m_j, s_j) | H_2(i, j) = 1 \text{ and } j = 1, \dots, n\} \end{aligned}$$

for $i = 1, \dots, k$.

3. Run $GT(x_{1,i}, 2^{k-1} - 1) = \sigma_i$ and $GT(x_{2,i}, 2^{k-1} - 1) = \sigma'_i$ for $i = 1, \dots, k$. Create two syndromes $\sigma = (\sigma_1, \dots, \sigma_k)$ and $\sigma' = (\sigma'_1, \dots, \sigma'_k)$.
4. Identify an index ℓ such that both $\sigma_\ell = 1$ and $\sigma'_\ell = 1$. As the two corresponding control groups complement each other and both are contaminated, this implies that each group contains a single bad signature.
5. Run the HV verifier for $x_{1,\ell}$ and identify the bad signature. In result, the index I_1 is known.
6. Calculate the second index $I_2 = I_1 \oplus \sigma \oplus \overline{\sigma'}$.
7. Run the GT test for the input batch without the two bad signatures identified by indices (I_1, I_2) . If the test accepts the batch, return the two indices, otherwise, the verifier fails and exits 1.

Take a closer look at the two-Layer Hamming Verifier (2HV). All steps are straightforward except the part when the second index is computed. Note that the indices and syndromes satisfy the following equations:

$$\begin{aligned} I_1 + I_2 &= \sigma \\ \overline{I_1} + \overline{I_2} &= \sigma' \end{aligned}$$

where $+$ is a bit-by-bit logical OR. Note that $+$ operation can be replaced by bit-by-bit XOR. Also the second equation can be converted using DeMorgan's Law. Thus

$$\begin{aligned} I_1 \oplus I_2 \oplus I_1 I_2 &= \sigma \\ I_1 I_2 &= \overline{\sigma'} \end{aligned}$$

This allows us to determine the other index knowing the first as

$$I_2 = I_1 \oplus \sigma \oplus \overline{\sigma'} \tag{13}$$

Let us analyse the complexity of the 2HV verifier. Step (1) takes one GT test. Step (3) consumes $2k$ GT tests. The HV verifier employed in Step (5) requires $(k - 1) + 2$ GT tests. Step (7) makes the final GT test. Overall, the 2HV verifier runs in expense of $3k + 3$ GT tests. So we can formulate the following conclusion.

Proposition 1. *Given a batch instance contaminated by two bad signatures. Then the 2HV verifier always correctly identifies them and consumes $3k + 3$ GT tests.*

Consider the case when instead of two bad signatures, a batch instance is contaminated by a single bad signature. The 2HV verifier will still work correctly returning two $I_1 = I_2$ indices. This case can be easily identified as syndromes $\sigma = \overline{\sigma'}$. This will allow to skip Step (5) and save on GT tests. If there is a high probability of a single bad signature occurring, then it would be better to run the HV verifier first (perhaps with $n = 2^k - 2$) and if it fails re-use the results in the 2HV verifier.

Now consider a simple example. Let a batch instance contain $n = 2^4 - 2 = 14$ signatures ($k = 4$). Assume that bad signature occur on 6th (0110) and 10th (1010) positions. The linear code used is defined by its matrix H of the form

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & \mathbf{0} & 1 & 0 & 1 & \mathbf{0} & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & \mathbf{1} & 1 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & \mathbf{0} & 0 & 1 & 1 & \mathbf{0} & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 \end{bmatrix}$$

We create sub-instances according to Step (2) and compute syndromes $\sigma = (0111)$ and $\sigma' = (1011)$. Note that $\sigma_3 = \sigma'_3$ so the third control groups in H_1 and H_2 complement each other and contain single bad signatures. Now we apply the HV verifier for the third control group in H_1 and identify $I_1 = (0110)$. The second index is $I_2 = I_1 \oplus \sigma \oplus \overline{\sigma'} = (0110) \oplus (0111) \oplus (0100) = (0101)$.

6 General Model for Verification Codes

Consider the 2HV verifier. One would hope that the indices I_1 and I_2 could be identified using $2k$ tests which correspond to the control groups defined by the matrix H . Ideally, one would expect that from the two equations

$$\begin{aligned} I_1 \oplus I_2 \oplus I_1 I_2 &= \sigma \\ f(I_1) \oplus f(I_2) \oplus f(I_1)f(I_2) &= \sigma', \end{aligned} \tag{14}$$

it is possible to determine both I_1 and I_2 . The function $f : \Sigma^k \rightarrow \Sigma^k$ is a Boolean function which for a given k -bit input, generates k -bit output. Now we prove that the following result is true.

Theorem 1. *Given four binary strings $I_1, I_2, \sigma, \sigma' \in \Sigma^k$ used in the 2HV verifier and satisfying Equation (14), then there is no function $f : \Sigma^k \rightarrow \Sigma^k$ for which the equations have unique solutions for I_1 and I_2 .*

Proof. First observe that Equation (14) is satisfied if and only if it is true for each bit. The proof reduces to the binary case – instead of Equation (14) we consider its binary version

$$\begin{aligned} i \oplus j \oplus ij &= u \\ f(i) \oplus f(j) \oplus f(i)f(j) &= v, \end{aligned} \tag{15}$$

where $i, j, u, v \in \Sigma$. For the function $f : \Sigma \rightarrow \Sigma$, there are four possibilities only: $f(x) \in \{0, 1, x, \bar{x}\}$. The constant functions $f(x) = 0$ and $f(x) = 1$ are not an option. The only candidates are $f(x) = x$ and $f(x) = \bar{x}$. The results are given in Table 3. Consider the value u (3rd column) and the value v for $f(x) = x$ (6th

Table 3. The truth table for two candidates of $f(x)$

i	j	u	$f(i) = i$	$f(j) = j$	v	$f(i) = \bar{i}$	$f(j) = \bar{j}$	v
0	0	0	0	0	1	1	1	1
0	1	1	0	1	1	0	1	1
1	0	1	1	0	0	1	1	1
1	1	1	1	1	0	0	0	0

column). If $u = v = 0$, there is the unique solution for $i = j = 0$. If $u = 0; v = 1$, there is no solution. For $u = v = 1$, there are three indistinguishable solutions. Consider the other function $f(x) = \bar{x}$ and the values u and v (9th column). If $u = 0$ and $v = 1$, there is unique solution $i = j = 0$. If $u = 1$ and $v = 1$, there is unique solution $i = j = 1$. For $u = v = 1$, there are two solutions $(i = 0, j = 1)$ and $(i = 1, j = 0)$. The combination $u = v = 0$ cannot occur.

Although the above theorem gives us a “cold” comfort, it also points towards a different approach. Given a batch instance of $n = 2^k$ signatures with $t = 2$ bad ones. We are looking for a matrix H with n columns and ℓ rows ($\ell > 2k$) such that any two indices I_1, I_2 (this time treated as the column vectors with ℓ bits) generate the unique result $I_1 + I_2$ (+ is bit-by-bit OR). In other words, we search for such an arrangement of rows of H that no two pairs of indices collide. The first question to be answered is the size of parameter ℓ for which a such arrangement may exist. If we assume that each column of the matrix H contains half of “1” then the parameter ℓ must satisfy the following inequality

$$\binom{\ell}{\frac{\ell}{2}} > \binom{n}{2}$$

It is easy to verify that for $k = 3$, $\ell \geq 2k + 1$. If k grows, then for $k = 40$, $\ell \geq 2k + 3$.

Definition 9. Generic Verifier (GV) *Given a batch instance x of length $n = 2^k$ for some positive k and a linear code represented by its parity check matrix*

H with ℓ rows ($\ell > 2k$) and n columns. Assume that the batch is contaminated by t bad signatures with their indices I_1, \dots, I_t which are column vectors of H . The syndrome $\sigma = I_1 + \dots + I_t$ which uniquely identifies the indices I_1, \dots, I_t .

1. Apply the generic test on the input instance. If $GT(x, n) = 0$, exit. Otherwise, go to the next step.
2. Create ℓ sub-instances (or control groups) corresponding to rows of the matrix H .
3. Run ℓ times the GT test and form the syndrome σ .
4. Identify indices I_1, \dots, I_t from the syndrome σ .
5. Run the GT test for the input batch without t bad signatures identified. If the test accepts the batch, return the t indices, otherwise, the verifier fails and exits 1.

6.1 Verification Codes from Combinatorial Designs

Combinatorial designs provide an inexhaustible source of structures with unlimited potential for new designs of verification codes. We start from a simple and not efficient structure to show at least, in principle, that verification codes may be constructed from well known combinatorial designs [7].

Theorem 2. Let D be the incidence matrix of a SBIBD(v, k, λ) where $v > 2k$, $k > 2\lambda$. Then D is a verification code allowing identification of any two bad signatures.

Proof. Note that columns in the D matrix represent the control groups or sub-collection of signatures which are to be tested. By contradiction. Assume that a 2-SBIBD has two pairs of rows (B_1, B_2) and (B_3, B_4) such that

$$B_1 \cup B_2 = B_3 \cup B_4.$$

Without loss of generality, we can write the incidence matrix of D where the first k elements of the first row B_1 are ones and the remainder are zeros. We can also write D with the first λ elements of the second row B_2 to be “1”, the next $k - \lambda$ elements to be “0”, the next $k - \lambda$ elements – “1” and the remaining k ones in the first $2k - \lambda$ – “0”. The next two rows $(B_3$ and $B_4)$ have k ones in the first $2k - \lambda$ columns and the last $v - 2k + \lambda$ elements zero since $B_1 \cup B_2 = B_3 \cup B_4$. Hence

$$D = \begin{bmatrix} \underbrace{1 \cdots 1 \cdots 1}_k & \underbrace{0 \cdots 0}_{k-\lambda} & \underbrace{0 \cdots 0}_{v-2k+\lambda} \\ \underbrace{1 \cdots 1}_\lambda & \underbrace{0 \cdots 0}_{k-\lambda} & \underbrace{1 \cdots 1}_{k-\lambda} & 0 \cdots 0 \\ \underbrace{k \text{ ones}}_{2k-\lambda} & & & 0 \cdots 0 \\ \underbrace{k \text{ ones}}_{2k-\lambda} & & & 0 \cdots 0 \end{bmatrix}$$

Consider the ones in the first three rows, ensuring the inner product is λ . Suppose that t ones overlap with both the first and second row ($0 \leq t \leq \lambda$). Since the last $(v - 2k + \lambda)$ columns contain only zeros, the number of ones in the third row is t in the first λ columns, $\lambda - t$ in the next $k - \lambda$ columns, and further $\lambda - t$ in the next $k - \lambda$ columns. Thus $t + 2\lambda - 2t = k$ and $t = 2\lambda - k \geq 0$. Hence $2\lambda \geq k$ – this is requested contradiction which proves the theorem.

7 Conclusions

Clearly, the above defined generic verifier sets the environment for the future research. In particular, the following list points out some open problems:

- lower bounds for the parameter ℓ or even better a function which determines the required parameter ℓ for a given t ,
- how to design the matrix H so the syndrome uniquely identifies the indices (bad signature positions),
- how to design a verification code so identification of bad signatures is efficient,
- determine t for which GV becomes no better than NV,
- constructions of verification codes from combinatorial designs.

Acknowledgement

The authors wish to thank anonymous referees for their critical comments.

References

1. M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT'98*, pages 236–250. Springer, 1998. Lecture Notes in Computer Science No. 1403. [29](#), [30](#)
2. M. Bellare and Y. Yacobi. Batch Diffie-Hellman key agreement systems and their application to portable communications. In R. Rueppel, editor, *Advances in Cryptology - EUROCRYPT'92*, pages 208–220. Springer, 1993. Lecture Notes in Computer Science No. 658. [29](#)
3. Elwyn Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, New York, 1968. [39](#)
4. J-S. Coron and D. Naccache. On the security of RSA screening. In H. Imai and Y. Zheng, editors, *Public Key Cryptography – Second International Workshop on Practice and Theory in Public Key Cryptography, PKC'99*, pages 197–203. Springer, 1999. Lecture Notes in Computer Science No. 1560. [29](#), [32](#)
5. L. Harn. Batch verifying multiple DSA-type digital signatures. *Electronics Letters*, 34(9):870–871, 1998. [30](#)
6. D. Naccache, D. M'Raihi, S. Vaudenay, and D. Raphaeli. Can DSA be improved ? complexity trade-offs with the digital signature standard. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT'94*, pages 77–85. Springer, 1995. Lecture Notes in Computer Science No. 950. [29](#)
7. A.P. Street and W.D. Wallis. *Combinatorics: A First Course*. CBRC, Winnipeg, 1982. [44](#)
8. S. Yen and C. Laih. Improved digital signature suitable for batch certification. *IEEE Transactions on Computers*, 44(7):957–959, 1995. [29](#), [31](#)

Some Remarks on a Fair Exchange Protocol

Jianying Zhou, Robert Deng, and Feng Bao

Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613
{jyzhou,deng,baofeng}@krndl.org.sg

Abstract. Fair exchange turns out to be an increasingly important topic due to the rapid growth of electronic commerce. An exchange is deemed to be fair if at the end of exchange, either each party receives the expected item or neither party receives any useful information about the other's item. Several protocols for fair exchange have been proposed in recent years. In this paper, we first examine a newly published fair exchange protocol and point out its flaws and weaknesses. We then put forward a more efficient and secure protocol and give an informal analysis.

Keywords: fair exchange, certified mail, secure electronic commerce

1 Introduction

Due to the rapid growth of electronic commerce nowadays, a related security issue on the fair exchange of electronic data between two parties over computer networks is of more and more importance. We can find various exchange instances in different types of commercial activity [2]:

- In contract signing, two parties exchange their non-repudiable commitment to the contract text.
- In purchasing, a payment is exchanged for a valuable item.
- In certified mail, a message is exchanged for an acknowledgement of receipt.

An exchange is *fair* if at the end of exchange, either each party receives the expected item or neither party receives any useful information about the other's item.

In electronic commerce scenarios, exchanges have to be carried over insecure networks and transacting parties may not trust each other. There could be subsequent disputes about what was exchanged during a transaction even if the exchange itself was completed fairly. In this case, evidence should be accumulated during the exchange to enable the settlement of any future disputes.

Solutions to the fair exchange problem reported in the literature fall into two categories:

- *Gradual exchange protocols* [4,5,6,8,12,13] where two parties gradually disclose the expected items by many steps.
- *Third party protocols* [1,2,3,7,9,11,14,15] which make use of an on-line or off-line (trusted) third party.

The gradual exchange solutions may have theoretical value but seem to be too cumbersome for actual implementation because of the high communication overhead. Hence, recent research mainly focuses on the third party solutions.

As the use of (trusted) third party *TTP* in fair exchange may cause the bottleneck problem, it is desirable to minimize the *TTP*'s involvement when designing efficient fair exchange protocols. Such an attempt has been made in [14], where the *TTP* acts as a *notary* rather than a delivery authority. However, the *TTP* still needs to be involved in each protocol run, though this might be necessary in some applications [15].

The *TTP*'s involvement is further reduced in [1,3,15], where transacting parties are willing to resolve communications problems between themselves and turn to the *TTP* only as a last recourse. However, only the risk-taking party (originator) is allowed to invoke the *TTP*, the responder may not know the final state of exchange in time. If a short time limit is imposed on a protocol run, the originator may not be quick enough to invoke the *TTP* for recovery thus the fairness will be destroyed.

Asokan, Shoup and Waidner proposed a generic fair exchange protocol in [2] which uses the *TTP* only in the case of exceptions and tolerates temporary failures in the communication channels to the *TTP*. In addition, it allows either party to unilaterally bring a protocol run to completion without losing fairness.

In this paper, we examine an instantiation of their generic fair exchange protocol for certified mail and put forward proposals for improvement. The following general notation is used throughout the paper.

- X, Y : concatenation of two messages X and Y .
- $H(X)$: a one-way hash function applied to message X .
- $eK(X)$ and $dK(X)$: encryption and decryption of message X with key K .
- $sS_A(X)$: principal A 's digital signature on message X with the private key S_A . The algorithm is assumed to be a ‘signature with appendix’, and the message is not recoverable from the signature.
- $A \rightarrow B : X$: principal A dispatches message X addressed to principal B .

2 ASW Protocol

A protocol for certified mail was proposed in [2] (see Figure 4 in the original paper). In this section, we give a brief description of the protocol, which is referred to as ASW protocol herein.

In certified mail, a sender O wants to send a mail message M to a receiver R . The sender O requires that the receiver R not be able to deny receiving the message M . To achieve this, O needs a non-repudiation of receipt token from R in exchange for the message M . Thus certified mail is a fair exchange of the message and its non-repudiation of receipt token.

ASW protocol has three sub-protocols: *exchange*, *abort*, and *resolve*. In the normal case, only the *exchange* sub-protocol is executed. The other two sub-protocols are used only if O or R presumes that something has gone wrong and decides to forcibly complete a protocol run. This is an indeterminate choice made *locally* by O or R without losing fairness. A (trusted) third party TTP will be invoked in the *abort* and *resolve* sub-protocols. It is assumed that communication channels between any two parties are *confidential*. It is further assumed that the communication channels between the TTP and each transacting party (O and R) are *resilient*, i.e. messages inserted into a resilient channel will eventually be delivered.

The notation below is used in the description of ASW protocol.

- P_{TTP} : the TTP 's public encryption key.
- V_O and V_R : verification keys of O and R respectively.
- key_O and key_R : random numbers generated by O and R respectively.
- $C = eP_{TTP}(M, key_O, V_O, V_R)$: encrypted mail message.
- $H(M)$: receipt text of a mail message M .

The *exchange* sub-protocol is as follows.

1. $O \rightarrow R : me1 = V_O, V_R, TTP, C, H(M), sS_O(V_O, V_R, TTP, C, H(M))$
IF R gives up **THEN** quit **ELSE**
2. $R \rightarrow O : me2 = H(key_R), sS_R(me1, H(key_R))$
IF O gives up **THEN** *abort* **ELSE**
3. $O \rightarrow R : me3 = M, key_O$
IF R gives up **THEN** *resolve* _{R} **ELSE**
4. $R \rightarrow O : me4 = key_R$
IF O gives up **THEN** *resolve* _{O}

The *abort* sub-protocol is as follows.

1. $O \rightarrow TTP : ma1 = aborted, me1, sS_O(aborted, me1)$
IF R has resolved **THEN** *resolve* _{O} **ELSE**
2. $TTP \rightarrow O : abort_token = ma1, sS_{TTP}(ma1)$

The *resolve* _{R} sub-protocol is as follows.

1. $R \rightarrow TTP : mrr1 = V_R, me1, me2, key_R$
IF $aborted$ **THEN**
2. $TTP \rightarrow R : mrr2 = abort_token$
ELSE
3. $TTP \rightarrow R : mrr3 = M, key_O$

The *resolve_O* sub-protocol is as follows.

1. $O \rightarrow TTP : mro1 = V_O, me1, me2, M, key_O$
IF aborted **THEN**
2. $TTP \rightarrow O : mro2 = abort_token$
ELSE
3. $TTP \rightarrow O : affidavit_token = affidavit, mro1, sS_{TTP}(affidavit, mro1)$

In the *exchange* sub-protocol, if R decides to give up before sending $me2$, it can simply terminate the protocol run without losing fairness. If O decides to give up after sending $me1$ (usually because O does not receive $me2$ within a reasonable time), it invokes the TTP by running the *abort* sub-protocol. If R decides to give up after sending $me2$ (typically because R does not receive $me3$ in time), it invokes the TTP by running the *resolve_R* sub-protocol. If O decides to give up after sending $me3$ (typically because O does not receive $me4$ in time), it invokes the TTP by running the *resolve_O* sub-protocol.

The *abort* sub-protocol is used by O to abort the protocol so that the TTP will not resolve the protocol at a later time. The *resolve_O* and *resolve_R* sub-protocols are used by O and R respectively to force a successful termination. Clearly, only one of the *abort* or *resolve* sub-protocols can succeed for a given instance of exchange. On the TTP 's system, each of the *abort* and *resolve* sub-protocol is guaranteed to be atomic.

In ASW protocol, either a tuple $(me1, me2, key_R)$ or an *affidavit_token* serves as a valid receipt for a mail message.

3 Some Remarks

The requirements for fair exchange were formulated in [2]:

- *Effectiveness*. If two parties behave correctly, they will receive the expected items without any involvement of the TTP .
- *Fairness*. After completion of a protocol run, either each party receives the expected item or neither party receives any useful information about the other's item.
- *Timeliness*. At any time during a protocol run, each party can unilaterally choose to terminate the protocol without losing fairness.
- *Non-repudiation*. If an item has been sent from party O to party R , O cannot deny origin of the item and R cannot deny receipt of the item.
- *Verifiability of Third Party*. If the third party misbehaves, resulting in the loss of fairness for a party, the victim can prove the fact in a dispute.

ASW protocol was designed to meet the above requirements. Nevertheless, some problems might exist.

Remark 1. *The abort sub-protocol is flawed.*

The *abort* sub-protocol is initiated by O , usually because O does not receive $me2$ in time. If R has already resolved the protocol, O is asked to initiate the *resolve_O* sub-protocol¹. However, O is unable to initiate the *resolve_O* sub-protocol without $me2$. Thus O has neither an *abort_token* nor an *affidavit_token* at the end of a protocol run while R has received the mail message.

In addition, O may misbehave by initiating the *abort* sub-protocol after it has initiated the *resolve_O* sub-protocol and obtained an *affidavit_token*. If the *TTP* sends an *abort_token* to O in this case, the *TTP* will be in a dilemma when R initiates the *resolve_R* sub-protocol later.

These problems also exist in their generic protocol for fair exchange. A fixed *abort* sub-protocol is as follows.

1. $O \rightarrow TTP : ma1 = \text{aborted}, me1, sS_O(\text{aborted}, me1)$
IF O has resolved **THEN**
2. $TTP \rightarrow O : ma2 = \text{affidavit_token}$
ELSE IF R has resolved **THEN**
3. $TTP \rightarrow O : ma3 = me2, key_R$
ELSE
4. $TTP \rightarrow O : \text{abort_token} = ma1, sS_{TTP}(ma1)$

Remark 2. *There is some redundancy in the resolve_O sub-protocol.*

O need not send M and key_O to the *TTP* in the *resolve_O* sub-protocol. With $me1$ and $me2$, it is sufficient for the *TTP* to issue the *affidavit_token*. If R resolves the protocol later, the *TTP* can obtain M by decrypting C contained in $me1$.

key_O was used in ASW protocol as a part of non-repudiation of origin token. In fact, $me1$ is already a complete non-repudiation of origin token. Hence, key_O can be omitted from all sub-protocols.

Remark 3. *The protocol performance may degrade when transmitting large mail messages.*

The *exchange* sub-protocol may become less efficient when the mail message is large since a mail message needs to be transmitted twice, that is cipher text C in $me1$ and plain text M in $me3$. The communication overheads will increase even more when the *abort* or *resolve* sub-protocols are invoked.

¹ Actually, R can initiate the *resolve_R* sub-protocol before sending $me2$ to O .

It is also a burden to the *TTP* to deal with the whole mail messages when the *abort* or *resolve* sub-protocols are invoked. The *TTP* needs a large space to store those mail messages and tokens safely until both parties have retrieved the expected items.

Remark 4. *The privacy of mail messages may not be well protected.*

As we just mentioned, if the *abort* or *resolve* sub-protocols are invoked, the content of a mail message has to be disclosed to the *TTP*. Such a situation may be undesirable to the parties who want to exchange mail messages secretly between themselves.

Although it is possible to encrypt the mail message either with a key shared between two parties or with the receiver's public encryption key before exchange, this will make the dispute resolution more complicated. If there is no evidence to prove what key is used and the times of encryption performed, the content of a mail message will be in dispute.

Remark 5. *The encrypted data in the non-repudiation of receipt token may not be publicly verifiable, which makes the dispute resolution inefficient.*

Suppose O sends the following $me1$ to R where $M' \neq M$.

$$me1 = V_O, V_R, TTP, C, H(M'), sS_O(V_O, V_R, TTP, C, H(M'))$$

If R does not receive $me3$ in time after sending $me2$, R may execute the *resolve-R* sub-protocol by sending $mrr1 = (V_R, me1, me2, key_R)$ to the *TTP*. If O has not aborted the protocol, the *TTP* will decrypt the mail message in $me1$ and send $mrr3 = (M, key_O)$ to R . If the *TTP* discloses key_R to O , O will have a complete receipt ($me1, me2, key_R$) which can make an arbitrator to believe that R received M' instead of M by only checking the receipt text $H(M')$. However, the *TTP*'s misbehavior cannot be verified since key_R could be sent to O by R itself after receiving M' from O .

This problem could be tackled if the arbitrator further checks whether the decrypted mail message M and the receipt text $H(M')$ in $me1$ match. If true, $me1$ is regarded as a valid part of receipt. However, this may require the *TTP*'s involvement because the mail message is encrypted with the *TTP*'s public key ². That means the *TTP* may need to be *on-line* for dispute resolution. If the *TTP* is temporary unavailable, arbitration has to be postponed. If the *TTP*'s private key has lost when a dispute arises, the arbitrator cannot make a proper conclusion.

² If the ElGamal public-key cryptosystem [10] is used, the encrypted message cannot be verified even with the plain message and the public key unless the random seed for ElGamal encryption is also provided. But the random seed is usually not saved after encryption.

The above problems may not be fatal to ASW protocol, but could affect its efficiency and security.

4 A Variant Protocol

Here we present a more efficient and secure protocol for certified mail, mainly based on the ideas from [2,14]. We split the definition of a mail message M into two parts, a commitment C and a key K . In the normal case, the originator O sends (C, K) (plus evidence of origin) to the recipient R in exchange for evidence of receipt without any involvement of the TTP . If there is something wrong in the middle of exchange, either O or R can unilaterally bring a protocol run to completion with the help from the TTP . The TTP only needs to notarise and/or deliver the message key K by request, which is usually much shorter than the whole mail message M .

The notation below is used in the description of our protocol.

- M : mail message being sent from O to R .
- K : message key defined by O .
- $C = eK(M)$: commitment (cipher text) for message M .
- $L = H(M, K)$: a unique label linking C and K .
- f_i ($1 \leq i \leq 8$): flags indicating the intended purpose of a signed message.
- $EOO_C = sS_O(f_1, R, L, C)$: evidence of origin of C .
- $EOR_C = sS_R(f_2, O, L, EOO_C)$: evidence of receipt of C .
- $EOO_K = sS_O(f_3, R, L, K)$: evidence of origin of K .
- $EOR_K = sS_R(f_4, O, L, EOO_K)$: evidence of receipt of K .
- $sub_K = sS_O(f_5, R, L, K, TTP, EOO_C)$: evidence of submission of K to the TTP .
- $con_K = sS_{TTP}(f_6, O, R, L, K)$: evidence of confirmation of K issued by the TTP .
- $abort = sS_{TTP}(f_8, O, R, L)$: evidence of abortion.
- P_{TTP} : the TTP 's public encryption key.

Like ASW protocol, our protocol has three sub-protocols: *exchange*, *abort*, and *resolve*. We also assume that the communication channels between the TTP and each transacting party (O and R) are *resilient*. In addition, we assume that the communication channel between O and R is *confidential* if the two parties want to exchange mail messages secretly. The *exchange* sub-protocol is as follows.

1. $O \rightarrow R : f_1, f_5, R, L, C, TTP, eP_{TTP}(K), EOO_C, sub_K$
IF R gives up **THEN** quit **ELSE**
2. $R \rightarrow O : f_2, O, L, EOR_C$
IF O gives up **THEN** *abort* **ELSE**
3. $O \rightarrow R : f_3, R, L, K, EOO_K$
IF R gives up **THEN** *resolve* **ELSE**
4. $R \rightarrow O : f_4, O, L, EOR_K$
IF O gives up **THEN** *resolve*

The *abort* sub-protocol is as follows.

1. $O \rightarrow TTP : f_7, R, L, sS_O(f_7, R, L)$
IF resolved **THEN**
2. $TTP \rightarrow O : f_2, f_6, O, R, L, K, con_K, EOR_C$
ELSE
3. $TTP \rightarrow O : f_8, O, R, L, abort$

The *resolve* sub-protocol is as follows, where the initiator U is either O or R .

1. $U \rightarrow TTP : f_2, f_5, O, R, L, TTP, eP_{TTP}(K), sub_K, EOO_C, EOR_C$
IF aborted **THEN**
2. $TTP \rightarrow U : f_8, O, R, L, abort$
ELSE
3. $TTP \rightarrow U : f_2, f_6, O, R, L, K, con_K, EOR_C$

In our protocol, evidence (EOR_C, EOR_K) or (EOR_C, con_K) can be used to prove that R received the message M , evidence (EOO_C, EOO_K) or (EOO_C, con_K) can be used to prove that O sent the message M .

If the *exchange* sub-protocol is executed successfully, R will receive C and K and thus $M = dK(C)$ together with non-repudiation of origin tokens (EOO_C, EOO_K) . Meanwhile, O will receive non-repudiation of receipt tokens (EOR_C, EOR_K) .

R can simply quit the transaction without losing fairness before sending EOR_C to O . Otherwise, R has to run the *resolve* sub-protocol to force a successful termination. Similarly, O can run the *abort* sub-protocol to quit the transaction without losing fairness before sending EOO_K to R . Otherwise, O has to run the *resolve* sub-protocol to force a successful termination.

The *resolve* sub-protocol can be initiated either by O or by R . When the TTP receives such a request, the TTP will first check the status of a transaction identified by (O, R, L) uniquely. If the transaction has been aborted by O , the TTP will return the *abort* token. If the transaction has already been resolved, the TTP will deliver the tuple $(f_2, f_6, O, R, L, K, con_K, EOR_C)$ to the current initiator of the *resolve* sub-protocol. Otherwise, the TTP will

- decrypt $eP_{TTP}(K)$ and verify with sub_K that K is submitted by O ;
- check that EOR_C is consistent with sub_K in terms of L and EOO_C ;
- generate evidence con_K ;
- deliver the tuple $(f_2, f_6, O, R, L, K, con_K, EOR_C)$ to the current initiator;
- set the status of the transaction *resolved*.

The third component in the tuple indicates the key supplier, which is authenticated by sub_K and notarised in con_K . Hence, intruders cannot mount a denial-of-service attack by sending bogus keys to the TTP for confirmation. Evidence con_K can be used to prove that

- a transaction identified by (O, R, L) has been resolved successfully,
- the message key K originated from O , and
- the message key K is available from the TTP by request.

In comparison with ASW protocol, our protocol has the following merits.

- The TTP 's overhead will not increase when transmitting large mail messages.
- The content of a mail message need not be disclosed to any outsiders including the TTP .
- The evidence is publicly verifiable without any restrictions on the types of signature and encryption algorithms.

Therefore, our protocol is more efficient and secure than ASW protocol both at the stage of exchange and at the stage of dispute resolution.

5 Security Analysis

We analyse our protocol with respect to the requirements listed in Section 3.

Claim 1. *If the communication channel between O and R is resilient, the protocol satisfies the effectiveness requirement.*

Proof: If both O and R are honest, they will send their messages according to the protocol description. If the communication channel between them is *resilient*, a message sent by either party will eventually be received by the other party. Thus the *exchange* sub-protocol can be executed successfully without invoking the TTP . R will receive C and K and thus $M = dK(C)$ together with non-repudiation of origin tokens (EOO_C , EOO_K). Meanwhile, O will receive non-repudiation of receipt tokens (EOR_C , EOR_K).

Claim 2. *If the communication channels between the TTP and each transacting party (O and R) are resilient, the protocol satisfies the fairness requirement.*

Proof: We first consider the possible unfair situations that O may face.

- O did not receive any message from R after sending message 1 in the *exchange* sub-protocol. In this case, O can initiate the *abort* sub-protocol, which is guaranteed to be completed within a finite period under the assumption. If R has not resolved the protocol, the TTP will not resolve the protocol at a later time, thus R cannot obtain K , which means R cannot receive M . If R has already resolved the protocol, O will obtain EOR_C and con_K from the TTP , which can be used to prove that R received M .
- O did not receive EOR_K after sending message 3 in the *exchange* sub-protocol. In this case, O can initiate the *resolve* sub-protocol to obtain con_K from the TTP , which can be used in place of EOR_K to prove that R received (or is able to receive) K provided the assumption holds.

The only possible unfair situation that R may face is that R did not receive K and EOO_K after sending message 2 in the *exchange* sub-protocol. In this case, R can initiate the *resolve* sub-protocol to obtain K and con_K under the same assumption.

Thus, the protocol satisfies the fairness requirement from both points of view of O and R .

Claim 3. *If the communication channels between the TTP and each transacting party (O and R) are resilient, the protocol satisfies the timeliness requirement.*

Proof: We first look at the possible ways that O can conclude a protocol run.

- terminating normally after sending message 4 in the *exchange* sub-protocol;
- invoking the *abort* sub-protocol at any time before sending message 3 in the *exchange* sub-protocol;
- invoking the *resolve* sub-protocol at any time after receiving message 2 in the *exchange* sub-protocol.

As we assume that the communication channel between the TTP and O is resilient, the *abort* and *resolve* sub-protocols initiated by O are guaranteed to be completed within a finite period. Thus at any time, there is always a way for O to conclude the protocol run.

On the other hand, the possible ways that R can conclude a protocol run are

- terminating normally after receiving message 4 in the *exchange* sub-protocol;
- simply quitting at any time before sending message 2 in the *exchange* sub-protocol;
- invoking the *resolve* sub-protocol at any time after receiving message 1 in the *exchange* sub-protocol.

Again, each of these will result in the timely conclusion of the protocol run for R .

Claim 4. *If the communication channels between the TTP and each transacting party (O and R) are resilient, the protocol meets the non-repudiation requirement.*

Proof: By the protocol description, R will hold the following non-repudiation of origin tokens

- (EOO_C, EOO_K) if the protocol terminates normally;
- (EOO_C, con_K) otherwise.

Meanwhile, O will hold the following non-repudiation of receipt tokens

- (EOR_C, EOR_K) if the protocol terminates normally;

- (*EOR-C*, *con-K*) otherwise.

EOO-C proves that O sent C with label L to R while *EOO-K* proves that O sent K with label L to R . Thus (*EOO-C*, *EOO-K*) proves that $M = dK(C)$ is from O . The link between C and K is computationally unique which should satisfy $L = H(dK(C), K)$.

In the same way, *EOR-C* proves that R received C with label L from O while *EOR-K* proves that R received K with label L from O . Thus (*EOR-C*, *EOR-K*) proves that R received $M = dK(C)$.

Alternatively, *con-K* proves that the *TTP* notarised K with label L at O 's request, and that R received (or is able to receive) K with label L from the *TTP* under the assumption of a resilient channel with the *TTP*. Thus *con-K* can be used with *EOO-C* and *EOR-C* to prove the origin and receipt of M .

Claim 5. *If the communication channels between the TTP and each transacting party (O and R) are resilient, and that the TTP can be forced to eventually send a valid response to any request sent to it, the TTP is verifiable.*

Proof: Under the assumptions, the *TTP*'s possible misbehavior could be

- R receives the *abort* token while O receives *con-K*;
- R receives K and *con-K* while O receives the *abort* token.

In the first case, if O uses *EOR-C* and *con-K* to prove that R received M , R can use the *abort* token to prove the *TTP*'s misbehavior.

In the second case, if R uses *EOO-C* and *con-K* to prove that O sent M to R , O can use the *abort* token to prove the *TTP*'s misbehavior. It should be noted that if R uses *EOO-C* and *EOO-K* to prove that O sent M to R , O cannot use the *abort* token to prove the *TTP*'s misbehavior since the *TTP* did not issue conflicting evidence.

6 Conclusion

We investigated ASW protocol and found out the following weaknesses besides some minor flaws being fixed.

- The performance may degrade when transmitting large mail messages.
- The privacy of mail messages may not be well protected.
- The *TTP* may need to be *on-line* for dispute resolution.

We proposed a variant protocol which overcomes the above weaknesses. The security analysis shows that our protocol meets the requirements for fair exchange.

References

1. N. Asokan, M. Schunter and M. Waidner. *Optimistic protocols for fair exchange*. Proceedings of 4th ACM Conference on Computer and Communications Security, pages 7–17, Zurich, Switzerland, April 1997. [47](#)
2. N. Asokan, V. Shoup and M. Waidner. *Asynchronous protocols for optimistic fair exchange*. Proceedings of 1998 IEEE Symposium on Security and Privacy, pages 86–99, Oakland, California, May 1998. [46, 47, 49, 52](#)
3. F. Bao, R. H. Deng and W. Mao. *Efficient and practical fair exchange protocols with off-line TTP*. Proceedings of 1998 IEEE Symposium on Security and Privacy, pages 77–85, Oakland, California, May 1998. [47](#)
4. M. Ben-Or, O. Goldreich, S. Micali and R. Rivest. *A fair protocol for signing contracts*. IEEE Transactions on Information Theory, IT-36(1):40–46, January 1990. [47](#)
5. E. F. Brickell, D. Chaum, I. B. Damgard and J. van de Graaf. *Gradual and verifiable release of a secret*. Lecture Notes in Computer Science 293, Advances in Cryptology: Proceedings of Crypto'87, pages 156–166, Santa Barbara, California, August 1987. [47](#)
6. R. Cleve. *Controlled gradual disclosure schemes for random bits and their applications*. Lecture Notes in Computer Science 435, Advances in Cryptology: Proceedings of Crypto'89, pages 573–588, Santa Barbara, California, August 1989. [47](#)
7. B. Cox, J. D. Tygar and M. Sirbu. *NetBill security and transaction protocol*. Proceedings of the First USENIX Workshop on Electronic Commerce, pages 77–88, July 1995. [47](#)
8. I. B. Damgard. *Practical and provably secure release of a secret and exchange of signatures*. Lecture Notes in Computer Science 765, Advances in Cryptology: Proceedings of Eurocrypt'93, pages 200–217, Lofthus, Norway, May 1993. [47](#)
9. R. H. Deng, L. Gong, A. A. Lazar and W. Wang. *Practical protocols for certified electronic mail*. Journal of Network and Systems Management, 4(3):279–297, 1996. [47](#)
10. T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithms*. IEEE Transactions on Information Theory, IT-31(4):469–472, July 1985. [51](#)
11. M. Franklin and M. Reiter. *Fair exchange with a semi-trusted third party*. Proceedings of 4th ACM Conference on Computer and Communications Security, pages 1–6, Zurich, Switzerland, April 1997. [47](#)
12. T. Okamoto and K. Ohta. *How to simultaneously exchange secrets by general assumptions*. Proceedings of 2nd ACM Conference on Computer and Communications Security, pages 184–192, Fairfax, Virginia, November 1994. [47](#)
13. P. Syverson. *Weakly secret bit commitment: Applications to lotteries and fair exchange*. Proceedings of 11th IEEE Computer Security Foundations Workshop, Rockport, Massachusetts, June 1998. [47](#)
14. J. Zhou and D. Gollmann. *A fair non-repudiation protocol*. Proceedings of 1996 IEEE Symposium on Security and Privacy, pages 55–61, Oakland, California, May 1996. [47, 52](#)
15. J. Zhou and D. Gollmann. *An efficient non-repudiation protocol*. Proceedings of 10th IEEE Computer Security Foundations Workshop, pages 126–132, Rockport, Massachusetts, June 1997. [47](#)

Gaudry's Variant against C_{ab} Curves

Seigo Arita

C&C Media Research Laboratories, NEC, Kawasaki Kanagawa, Japan,
`arita@ccm.cl.nec.co.jp`

Abstract. Gaudry has described a new algorithm (Gaudry's variant) for the discrete logarithm problem (DLP) in hyperelliptic curves. For hyperelliptic curves of small genus on finite field $\text{GF}(q)$, Gaudry's variant solves for the DLP in $O(q^2 \log^\gamma(q))$ time. This paper shows that C_{ab} curves can be attacked with a modified form of Gaudry's variant and presents the timing results of such attack. However, Gaudry's variant cannot be effective in all of the C_{ab} curve cryptosystems, this paper provides an example of a C_{ab} curve that is unassailable by Gaudry's variant.

1 Introduction

Gaudry has described a new algorithm (Gaudry's variant) for the discrete logarithm problem (DLP) in hyperelliptic curves [7]. Gaudry's variant uses the method for Pollard's rho algorithm [12] with the function field sieving algorithm of Adleman, DeMarrais, and Huang [1]. Gaudry's variant solves the DLP in hyperelliptic curves of genus g defined on the finite field F_q in time $O(q^2 \log^\gamma(q))$ when the genus g is sufficiently small in comparison to the order q of the definition field.

Arita and Galbraith et al. have described addition algorithms on the Jacobian group of C_{ab} and superelliptic curves respectively, and have demonstrated algorithm applications in discrete-log-based public key cryptosystems [3,6]. This paper shows that C_{ab} and superelliptic curves can be attacked by a modified Gaudry's variant and presents timing results from the attack.

With hyperelliptic or C_{ab} curve cryptosystems, researchers usually select a sufficiently large genus so that definition fields are less than one word in size to hasten computations [14,3]. Gaudry's variant has excluded out this conventional hastening method. However, Gaudry's variant cannot be effective in all of the non-elliptic algebraic curve cryptosystems. This paper provides an example of a C_{ab} curve that is unassailable by Gaudry's variant.

2 Gaudry's Variant

Take hyperelliptic curve $C : y^2 = x^{2g+1} + a_1x^{2g} + \cdots + a_{2g+1}$ of genus g defined on finite field F_q . Suppose the genus g is sufficiently small in comparison to the order q of the definition field. Let J_C denote the Jacobian group of the hyperelliptic

curve. To handle with the DLP, look for integer λ that satisfies $D_2 = \lambda D_1$ for two elements D_1 and D_2 in J_C .

In Pollard's rho algorithm, we calculate the random linear sums $R_i = \alpha_i D_1 + \beta_i D_2$ ($i = 1, 2, \dots$) of D_1 and D_2 step by step through a random walk, and wait for a collision $R_i = R_j$. Once it occurs, from $\alpha_i D_1 + \beta_i D_2 = \alpha_j D_1 + \beta_j D_2$, we get $D_2 = (\alpha_i - \alpha_j)/(\beta_j - \beta_i)D_1$, and λ is obtained.

In Gaudry's variant, just as in the case of rho algorithm, we calculate the linear sums $R_i = \alpha_i D_1 + \beta_i D_2$ ($i = 1, 2, \dots$) of D_1 and D_2 step by step through a random walk. However, we gather smooth R_i value's instead of waiting for a collision. Element D in J_C is called smooth when D is the sum of F_q rational points on C ; that is, we take all of the F_q rational points as a factor base. Let all of the F_q rational points on C be $\{P_1, P_2, \dots, P_w\}$. Then, w -dimensional vector $M_i = (m_{i,1}, \dots, m_{i,w})$ corresponds to every smooth R_i through $R_i = \sum_k m_{i,k} P_k$. In the polynomial expression $R_i = [u_i(x), v_i(x)]$ from Cantor's algorithm [4], R_i is smooth if and only if the polynomial $u_i(x)$ is factored into the linear product $\prod_k (x - c_k)$ over F_q , and then we get $R_i = \sum_k (c_k, v(c_k))$. Therefore, by calculating all of the F_q rational points $\{P_1, P_2, \dots, P_w\}$ in advance at a complexity of $O(q)$, vector M_i is easily obtained.

When g is sufficiently small in comparison to q , about $1/g!$ of all of the elements in J_C are smooth ([7]Prop.4). We encounter a smooth R_i for every $g!$ steps (remember g is small), and we obtain w' ($\geq w$) smooth R_i values after $g! \cdot w'$ steps. The w' vectors $M_i = (m_{i,1}, \dots, m_{i,w})$ ($i = 1, \dots, w'$) then become linearly dependent, and by solving the w -dimensional linear equation, we obtain values for γ_i ($i = 1, \dots, w'$) that satisfy

$$\sum_{i=1}^{w'} \gamma_i M_i = 0.$$

These values for γ_i produce λ through the equation $R_i = \sum_k m_{i,k} P_k$;

$$\lambda = - \sum_i \gamma_i \alpha_i / \sum_i \gamma_i \beta_i.$$

In calculating Gaudry's variant, the most complex step is that of solving the w -dimensional linear equation $\sum_i \gamma_i M_i = 0$ ($i = 1, 2, \dots, w'$). The matrix $(m_{i,k})$ has a size equal to about $q \times q$ and is sparse since there are only g non-zero elements in each row; thus the linear equation can be solved in q^2 steps. The complexity of Gaudry's variant is given as $O(q^2 \log^\gamma(q))$.

When C has non-trivial automorphism ϕ , Gaudry's variant becomes more powerful. Let m denote the order of ϕ . In this case, all of the F_q rational points are unnecessary. Only the representatives of orbits in F_q rational points for the action of ϕ are needed as a factor base. The number of the orbits is q/m , so the complexity of Gaudry's variant becomes $O((q/m)^2 \log^\gamma(q))$.

However, automorphisms in C can be ignored, according to a theorem from Arbarello et al [2].

Theorem 1. *The order of the automorphism group of a smooth algebraic curve of genus ≥ 2 is at most $84(g - 1)$.*

Using this theorem, we know that the effect of automorphisms can be ignored by expanding the size of the definition field with $\log_2(84(g - 1))$ more bits.

3 C_{ab} and Superelliptic Curve

The C_{ab} curve is a nonsingular affine curve with the equation

$$\sum_{0 \leq i \leq b, 0 \leq j \leq a, ai + bj \leq ab} \alpha_{i,j} x^i y^j = 0,$$

where both $\alpha_{b,0}$ and $\alpha_{0,a}$ are not equal to zero [10]. Arita has described an addition algorithm for the Jacobian group of a C_{ab} curve in terms of ideals of the coordinate ring; he has also proposed discrete-log-based public key cryptosystems using C_{ab} curves [3]. For a C_{ab} curve with 160 bits of a Jacobian group, timing results from the addition algorithm are listed in Tables 1, 2, and 3.

Table 1. Performance for the C_{35} curve in ms at 266 MHz, using a Pentium II chip.

	simple	random
sum	3.39	3.65
double	3.76	4.21
scalar	862	958

Table 2. Performance for the C_{37} curve in ms at 266 MHz, using a Pentium II chip.

	simple	random
sum	1.15	1.24
double	1.15	1.28
scalar	273	300

Table 3. Performance for the $C_{2,13}$ curve in ms at 266 MHz, using a Pentium II chip.

	simple	random
sum	0.70	0.73
double	0.65	0.68
scalar	158	167

In the tables, “simple” denotes C_{ab} curves with equation $Y^a + \alpha X^b + \beta$, and “random” denotes randomly chosen C_{ab} curves. “Sum”, “double”, and “scalar”

denote addition, doubling, and scalar multiplication of random elements, respectively.

A superelliptic curve is a nonsingular affine curve of the equation

$$y^n = a_\delta x^\delta + \cdots + a_0,$$

where n is prime to the characteristics of the definition field, and n and δ are prime to each other [6]. Galbraith et al. have described an addition algorithm in the Jacobian group of a superelliptic curve in terms of lattice computation [6].

Given these definitions, C_{ab} curves clearly include superelliptic curves. Therefore, only C_{ab} curves will be examined.

4 Application of Gaudry's Variant to C_{ab} Curves

I modified Gaudry's variant and applied it to C_{ab} curves. The problems I encountered were how to decide if a given element in a Jacobian group is smooth or not, and, if it is smooth, how to represent the element as a sum of F_q rational points.

For example, with a C_{37} curve, where the genus is 6, element R in the Jacobian group is expressed as an ideal of the coordinate ring using the Gröbner basis with respect to the C_{37} order [3]:

$$\begin{aligned} R = & \{a_0 + a_1x + a_2x^2 + a_3y + a_4x^3 + a_5xy + x^4, \\ & b_0 + b_1x + b_2x^2 + b_3y + b_4x^3 + b_5xy + x^2y, \\ & c_0 + c_1x + c_2x^2 + c_3y + c_4x^3 + c_5xy + y^2\}. \end{aligned}$$

Here, a_i , b_i , and c_i are elements in the definition field. The common zeroes of these three equations are six points on the C_{37} curve, that comprise R . When definition field F_q is large enough, for almost any R in the Jacobian group, the six points comprising R have distinct x -coordinates to each other. So, almost any R can be expressed as common zeroes of two polynomials:

$$\begin{aligned} R = & \{\text{sixth degree polynomial of } x, \\ & y + (\text{fifth degree polynomial of } x)\}, \end{aligned}$$

just as in hyperelliptic curves. The expression is nothing but the Grobner basis of (the ideal corresponding to) R with respect to the lexicographic order.

Therefore, in Gaudry's variant against C_{ab} curves, for almost any R_i , the decision regarding the R_i value's smoothness and representation as a sum of F_q rational points follows the same pattern as for hyperelliptic curves, after translating the Gröbner basis of R_i w.r.t. C_{ab} order to another Gröbner basis w.r.t. lexicographic order. It is well known that Gröbner bases can be effectively translated between distinct monomial orders. If exceptional values for R_i are found, they are simply discarded. It is known that Gaudry's variant can also be applied to C_{ab} curves.

Table 4. 93 bits of a C_{37} curve over 17 bits of a prime field

finite field	F_{84211}
defining equation	$1 + 24740x^7 + 32427y^3 = 0$
genus	6
order of Jacobian	$43 \cdot 8068970623016239605318986617$
order of automorphism	$3 \cdot 7$

I implemented the modified Gaudry's variant with C language and the PARI-GP [11], and then tested it against the C_{37} curve in Table 4. The curve has 93 bits of a Jacobian group over 17 bits of a prime field. Let ζ_3 and ζ_7 denote the primitive seventh root of one and the third root of one, respectively. Since the curve has the automorphism $\phi(x, y) = (\zeta_7 x, \zeta_3 y)$ of order 21, the number of rational points in a factor base should be $84211/21 = 4010 \dots$ or more. I needed to collect 4011 or more smooth elements and then solve about 4011 dimensional spare linear equations. For solving these linear equations, I used the Lanczos algorithm [9].

I randomly generated two elements, D_1 and D_2 , in the Jacobian group:

$$D_1 = \{x^4 + 77465x^3 + 75875x^2 + (37117y + 57992)x$$

$$+ (42876y + 21588),$$

$$5485x^3 + (y + 79222)x^2 + (4298y + 50456)x$$

$$+ (36882y + 81869),$$

$$41971x^3 + 64608x^2 + (26263y + 16207)x$$

$$+ (y^2 + 42778y + 62216)\},$$

$$D_2 = \{x^4 + 64296x^3 + 44620x^2 + (29434y + 15779)x$$

$$+ (61013y + 42557),$$

$$51156 * x^3 + (y + 32172)x^2 + (62401y + 22153)x$$

$$+ (78055y + 13056),$$

$$79116x^3 + 5028x^2 + (69977y + 21979)x$$

$$+ (y^2 + 75761y + 2009)\}.$$

Then, running the modified Gaudry's variant, I obtained

$$\lambda = 4082271804134874346983670415$$

for $D_2 = \lambda D_1$ in the time given in Table 5. The average and variance of the number of steps needed to produce every smooth element were 848.265 and 680786, respectively, reasonable results when compared to Gaudry's theoretical estimate of $6! = 720$ [7].

Table 5. Timing results of modified Gaudry's variant against the C_{37} curve from Table 4 using a 266-MHz Pentium II chip.

Collection of rational points (PARI-GP)	5 m 13 s
Collection of smooth elements (C)	2 h 33 m 6 s
Solving linear equation (C)	32 m 2 s
Total	3 h 11 m 21 s

5 C_{ab} Curves that are Unassailable by Gaudry's Variant

Let a and b be distinct prime numbers. Let $C(p, \alpha, \beta) (= C(p, a, b, \alpha, \beta))$ denote a C_{ab} curve over prime field F_p with the equation

$$\alpha Y^a + \beta X^b + 1 = 0.$$

In this section, I will construct a C_{ab} curve $C(p, \alpha, \beta)$ of a small genus, that is secure against Gaudry's variant. Let h be the order of the Jacobian group of $C(p, \alpha, \beta)$. Conditions for security are:

Condition 1 h has at least 160 bits of prime factor l [12],

Condition 2 prime factor l does not divide $p^k - 1$ for small values of k [5],

Condition 3 prime factor l is not equal to p [13], and

Condition 4 p has $(40 + \log_2(84(g - 1)))$ or more bits.

The fourth condition secures a curve against Gaudry's variant. Note that the effect of automorphisms can be ignored by expanding the size of the definition field with $\log_2(84(g - 1))$ more bits (refer to Theorem 1).

Because we are handling a curve with a small genus, we do not need to consider Adleman, DeMarrais, and Huang's algorithm [1] or its extension to superelliptic curves [6].

Koblitz used Jacobi sums to calculate the order of Jacobian groups of hyperelliptic curves [8]. Jacobi sums can also be used for curve $C(p, \alpha, \beta)$. For simplicity, $p \equiv 1 \pmod{\text{lcm}(a, b)}$ has been assumed.

Fix the generator w of multiplicative group F_p^* . For a rational number s where $(p - 1)s$ is an integer, character χ_s of F_p^* is defined by

$$\chi_s(w) = e^{2\pi i s}.$$

Then, extend χ_s to the whole F_p by setting $\chi_s(0) = 0$ when s is not an integer and setting $\chi_s(0) = 1$ when it is.

For integers $l = 1, 2, \dots, a - 1$ and $m = 1, 2, \dots, b - 1$,

$$j_p(l, m) = \sum_{1+v_1+v_2=0} \chi_{l/a}(v_1) \chi_{m/b}(v_2) \quad (1)$$

is called a Jacobi sum. Here, v_1 and v_2 run over F_p under the condition $1 + v_1 + v_2 = 0$.

Weil has demonstrated that the L function $L_p(U)$ of $C(p, \alpha, \beta)$ can be expressed by Jacobi sums [15]:

$$L_p(U) = \prod_{\substack{l=1,2,\dots,a-1 \\ m=1,2,\dots,b-1}} (1 + \overline{\chi_{l/a}}(\alpha) \overline{\chi_{m/b}}(\beta) j_p(l, m) U),$$

where $\overline{\chi_s}$ denotes the complex conjugate of χ_s .

Generally, the order of a Jacobian group of a curve is equal to the value of the L function $L(U)$ of the curve at $U=1$, so the order h of the Jacobian group of $C(p, \alpha, \beta)$ is given as

$$\begin{aligned} h &= L_p(1) \\ &= \prod_{\substack{l=1,2,\dots,a-1 \\ m=1,2,\dots,b-1}} (1 + \overline{\chi_{l/a}}(\alpha) \overline{\chi_{m/b}}(\beta) j_p(l, m)) \end{aligned} \tag{2}$$

Thus, to know the order h of the Jacobian group, it is sufficient to calculate the Jacobi sums $j_p(l, m)$. However, they cannot be calculated directly using the formula (1) for Jacobi sums, so we use the Stickelberger element to calculate them.

Let $[\lambda]$ denote the largest integer under the rational number λ , and $<\lambda>$ denote $\lambda - [\lambda]$. Take cyclotomic field $\mathbb{Q}(\zeta)$ with a primitive ab -th root ζ of 1. Let σ_t denote the Galois map $\zeta \mapsto \zeta^t$ of $\mathbb{Q}(\zeta)$. An element $\omega(a, b)$ in group ring $\mathbb{Z}[Gal(\mathbb{Q}(\zeta)|\mathbb{Q})]$ defined by

$$\omega(a, b) = \sum_t \left[<\frac{t}{a}> + <\frac{t}{b}> \right] \sigma_{-t}^{-1}, \tag{3}$$

where t runs over reduced residue classes mod ab , is called a Stickelberger element. As an ideal of $\mathbb{Q}(\zeta)$,

$$(j_p(l, m)) = P^{\omega(a, b)}, \tag{4}$$

where P denotes an prime ideal of $\mathbb{Q}(\zeta)$ lying over p [16]. By knowing the prime p and the prime ideal P in advance, Equation (4) can be used to determine $j_p(l, m)$ up to the power of $-\zeta$.

By determining the Jacobi sums using a Stickelberger element, the following algorithms can be obtained:

Algorithm 1 (To search for a secure $C(p, \alpha, \beta)$)

Input: a, b

Output: p, α, β , and the order h of the Jacobian group

1. $g \leftarrow (a - 1)(b - 1)/2$
 2. $m \leftarrow \text{Max}(\lceil 160/g \rceil, \lceil 40 + \log_2(84(g - 1)) \rceil)$
- $\lceil n \rceil$ denotes the least integer over n .

3. Search the candidate j for value of a Jacobi sum for some prime p of m or more bits by using Algorithm 2:

$$(p, j) \leftarrow \text{Algorithm 2}(m).$$

4. For every $k = 0, 1, \dots, ab - 1$,

$$h_k \leftarrow \prod_{\substack{l=1,2,\dots,a-1 \\ m=1,2,\dots,b-1}} (1 + (-\zeta)^k j).$$

5. Check that there is a value h_k that satisfies Conditions 1, 2, and 3 for security in the set $\{h_0, h_1, \dots, h_{ab-1}\}$. If there is not, Go to step 3. If it does, $h \leftarrow h_k$.
6. Let ζ_a and ζ_b denote the primitive a -th and b -th root of 1, respectively. For every $l = 0, 1, \dots, a - 1$ and $m = 0, 1, \dots, b - 1$, check if the order of the Jacobian group of $C(p, \zeta_a^l, \zeta_b^m)$ is equal to h or not; for example, check if h times the random element is equal to the unit element, or not. If the order is equal, output $p, \alpha = \zeta_a^l, \beta = \zeta_b^m$, and h . If there is no such l and m , go to step 3.

As per step 2, p has $40 + \log_2(84(g - 1))$ or more bits, so the curve obtained by Algorithm 1 is secure against Gaudry's variant.

Algorithm 2, contained in Algorithm 1, makes use of Equation (3) and (4) for the Stickelberger element to find the candidate value of the Jacobi sum.

Algorithm 2 (To find the candidate of the Jacobi sum)

Input: m

Output: p and j

1. $\omega \leftarrow \sum_t (<\frac{t}{a}> + <\frac{t}{b}>) \sigma_{-t}^{-1}$
2. Randomly generate $\gamma_0 = \sum_{l=0}^{(a-1)(b-1)-1} c_l \zeta^l$ ($-20 \leq c_l \leq 20$).
3. For every $i = 1, 2, \dots,$

$$\gamma \leftarrow \gamma_0 + i$$

$$p \leftarrow \text{Norm}_{\mathbb{Q}(\zeta)|\mathbb{Q}}(\gamma)$$

If $p < 2^m$, then try the next i .

If $p >> 2^m$, then go to step 2.

' $>>$ ' means 'sufficiently larger than.'

If p is not prime, then try the next i .

4. $j \leftarrow \gamma^\omega$

Output p and j .

Example Algorithm 1 was run for $a = 3$ and $b = 5$.

1. $g \leftarrow (3 - 1)(5 - 1)/2 = 4$
2. $m \leftarrow \max(160/4, \lceil 40 + \log_2(84 \cdot 3) \rceil) = 48$
3. Run Algorithm 2 for $m = 48$.
 - (a) $\omega \leftarrow \sum_t (<\frac{t}{3}> + <\frac{t}{5}>) \sigma_{-t}^{-1} = \sigma_1 + \sigma_7 + \sigma_{11} + \sigma_{13}$

- (b) $\gamma_0 \leftarrow 20 - 3\zeta - 12\zeta^2 + 20\zeta^3 - 11\zeta^4 - 4\zeta^5 + 3\zeta^6 - 16\zeta^7$ was randomly generated.

For $\gamma \leftarrow \gamma_0 + 60 = 80 - 3\zeta - 12\zeta^2 + 20\zeta^3 - 11\zeta^4 - 4\zeta^5 + 3\zeta^6 - 16\zeta^7$,
 $p \leftarrow \text{Norm}(\gamma) = 581929936583251$ is a prime of 50 bits. Now, the Jacobi sum candidate j for p is

$$\begin{aligned} j &\leftarrow \gamma\gamma^{(7)}\gamma^{(11)}\gamma^{(13)} \\ &= 18682331 + 1900434\zeta + 3903200\zeta^2 \\ &\quad + 735220\zeta^3 + 2683534\zeta^4 - 6028054\zeta^5 \\ &\quad - 1372490\zeta^6 + 3103044\zeta^7 \end{aligned}$$

4. For every $k = 0, 1, \dots, 29$, compute $h_k \leftarrow \text{Norm}(1 + (-\zeta)^k j)$:

$$h_0 \leftarrow 114678672941303554807554279710671283827257404103736047882496$$

...

$$h_6 \leftarrow 114678699138696308587273958663265811323988422727826915122881$$

...

$$h_{29} \leftarrow 114678746421612909844326492247007547650638094985955354652416$$

5. h_6 satisfies Condition 1,2 and 3. In fact,

$$h \leftarrow h_6 = 2511 \cdot l$$

Here,

$$l = 45670529326442177852359202972228519045793876036569858671$$

is a prime of 185 bits, distinct from p . Condition 2 is satisfied for $k \leq 1000$.

6. For $\alpha = 579364850535396$ and $\beta = 289406374935593$, it is verified that the order of the Jacobian group of $C(p, \alpha, \beta)$ is equal to h .

Thus, a C_{35} curve is obtained with

$$579364850535396y^3 + 289406374935593x^5 + 1 = 0$$

over the prime field $\text{GF}(581929936583251)$ with the Jacobian group of the order

$$2511 \cdot 45670529326442177852359202972228519045793876036569858671,$$

which is secure against Gaudry's variant.

6 Conclusion

I have demonstrated that C_{ab} and superelliptic curves can be attacked by a modified Gaudry's variant, and I reported timing results for the attack. Miura has demonstrated that any algebraic curve with at least one rational point has a C_{ab} -type model [10]. Therefore, I determined that Gaudry's variant could be applied to virtually all algebraic curves.

However, Gaudry's variant cannot be effective in all non-elliptic algebraic curve cryptosystems. I have provided an example of a C_{35} curve that is unavailable by Gaudry's variant.

References

1. L.M.Adleman,J.DeMarrais, and M.D.Huang, "A Subexponential Algorithm for Discrete Logarithms over the Rational Subgroup of the Jacobians of Large Genus Hyperelliptic Curves over Finite Fields," ANTS-I, LNCS 877, Springer, 1994. [58](#), [63](#)
2. E.Arbarello, M.Cornalba, P.A.Griffiths, and J.Harris, "Geometry of Algebraic Curves Volume I," Springer-Verlag, 1984. [59](#)
3. S. Arita, "Algorithms for computations in Jacobian group of C_{ab} curve and their application to discrete-log-based public key cryptosystems," Conference on The Mathematics of Public Key Cryptography, Toronto, 1999. [58](#), [60](#), [61](#)
4. D.G.Cantor, "Computing in the Jacobian of a hyperelliptic curve," Mathematics of Computation, 48(177), pp.95-101,1987. [59](#)
5. G.Frey and H.-G.Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," Math. Comp.,62(206),pp.865-874,1994. [63](#)
6. S.D.Galbraith, S.Paulus, and N.P.Smart "Arithmetic on Superelliptic Curves," preprint,1999. [58](#), [61](#), [63](#)
7. P.Gaudry, "A variant of the Adleman-DeMarris-Huang algorithm and its application to small genera," Conference on The Mathematics of Public Key Cryptography, Toronto, 1999. [58](#), [59](#), [62](#)
8. N.Koblitz, "A very easy way to generate curves over prime fields for hyperelliptic cryptosystems," Rump-session Crypto'97, 1997. [63](#)
9. B.A.LaMacchia and A.M.Odlyzko, "Solving large sparse linear systems over finite fields," Crypto '90, LNCS 537, Springer, 1990. [62](#)
10. S. Miura, "Linear Codes on Affine Algebraic Curves," Transactions of IEICE, Vol. J81-A, No 10, pp.1398-1421,1998. [60](#), [66](#)
11. PARI-GP, <ftp://megrez.math.u-bordeaux.fr/pub/pari> [62](#)
12. J.M.Pollard, "Monte Carlo methods for index computation mod p," Math. Comp.,32(143),pp.918-924,1978. [58](#), [63](#)
13. H.-G.Rück, "On the discrete logarithm in the divisor class group of curves," Math. Comp.,68(226),pp.805-806,1999. [63](#)
14. Y.Sakai and K.Sakurai, "Design of hyperelliptic cryptosystems in small characteristic and a software implementation over F_{2^n} ," Asiacrypt '98, Advances in Cryptology, LNCS 1514, Springer, 1998. [58](#)
15. A.Weil "Numbers of solutions of equations in finite fields," Bull.Amer.Math.Soc.,55,pp.497-508,1949. [64](#)
16. A.Weil "Jacobi Sums as "Größencharaktere"," Trans.Amer.Math.Soc.,73,pp.487-495,1952. [64](#)

An Identification Scheme Based on Sparse Polynomials

William D. Banks¹, Daniel Lieman², and Igor E. Shparlinski³

¹ Department of Mathematics, University of Missouri
Columbia, MO 65211, USA
bbanks@math.missouri.edu

² Department of Mathematics, University of Missouri
Columbia, MO 65211, USA
lieman@math.missouri.edu

³ Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
igor@mpce.mq.edu.au

Abstract. This paper gives a new example of exploiting the idea of using polynomials with restricted coefficients over finite fields and rings to construct reliable cryptosystems and identification schemes.

1 Overview

The recently discovered idea of using polynomials with restricted coefficients in cryptography has already found several cryptographic applications such as the NTRU cryptosystem [7], the ENROOT cryptosystem [4] and the PASS identification scheme [6]; see also [5].

In contrast to the constructions of NTRU and PASS, which consider classes of polynomials of low degree with many “small” non-zero coefficients, ENROOT introduced a public key cryptosystem where the polynomials are of high degree, but extremely sparse. In this paper, we give a new application of this idea to the design of a fast and reliable identification scheme.

Let q be a prime power and let \mathbb{F}_q be the finite field of q elements.

Given a set $S \subseteq \mathbb{F}_q$, we say that a polynomial $G(X) \in \mathbb{F}_q[X]$ is an *S-polynomial* if every coefficient of G belongs to S , and we say that it is an *essentially S-polynomial* if $G(X) - G(0)$ is an *S-polynomial*. (This notation is somewhat reminiscent of the idea of *S-units* in number theory, and is not related to constructions in algebraic geometry.)

Finally, we say that a polynomial $G(X) \in \mathbb{F}_q[X]$ is τ -sparse if it has at most τ non-zero coefficients.

Throughout this paper $\log z$ denotes the binary logarithm of z .

The “hard” problem underlying our one-way functions can be stated as follows:

Given $2m$ arbitrary elements $\alpha_1, \dots, \alpha_m, \gamma_1, \dots, \gamma_m \in \mathbb{F}_q$ and a set $S \subseteq \mathbb{F}_q$ of small cardinality, it is unfeasible to find a τ -sparse S -polynomial $G(X) \in \mathbb{F}_q[X]$ of degree $\deg G \leq q - 1$ such that $G(\alpha_j) = \gamma_j$ for $j = 1, \dots, m$, provided that q is of “medium” size relative to the choice of $m \geq 1$ and $\tau \geq 3$.

More precisely, we expect that if one fixes the number of points m , the cardinality $|S|$ and the sparsity $\tau \geq 3$, then the problem requires exponential time as $q \rightarrow \infty$. Of course, we mean exponential time in the bit length of q , that is, in $\log q$.

Indeed, consider the special case $q = p$, where p is a prime number. Let $a_{ij} \equiv \alpha_j^i \pmod{p}$ and $c_j \equiv \gamma_j \pmod{p}$ be chosen so that $0 \leq a_{ij}, c_j \leq p - 1$ for $i = 0, \dots, p - 1$ and $j = 1, \dots, m$. In this (simplified!) case, the hard problem above is still equivalent to the hard problem of finding a feasible solution to the *integer programming problem*

$$\sum_{i=0}^{p-1} x_i \varepsilon_i a_{ij} + y_j p = c_j, \quad j = 1, \dots, m, \quad \sum_{i=0}^{p-1} \varepsilon_i \leq \tau,$$

where $y_j \in \mathbb{Z}$, $x_i \in S$, and $\varepsilon_i \in \{0, 1\}$ for all i and j .

2 Basic Idea

Let us fix the finite field \mathbb{F}_q and some integer parameters $k \geq 1$ and $r, s, t \geq 2$. To create the signature *Alice* uses the following algorithm – which we denote SPIFI, for Secure Polynomial IdentifiCation.

Initial Set-up

Step 1

Select at random k distinct elements $a_0, \dots, a_{k-1} \in \mathbb{F}_q$ and a random t -sparse $\{0, 1\}$ -polynomial $\varphi(X) \in \mathbb{F}_q[X]$ of degree at most $q - 1$ and with $\varphi(0) = 0$.

Step 2

Compute $A = -\varphi(a_0)$, and put $f(X) = \varphi(X) + A$. Thus f is a t -sparse essentially $\{0, 1\}$ -polynomial with $f(a_0) = 0$ and $f(0) = A$.

Step 3

Compute $C_j = f(a_j)$, $j = 1, \dots, k - 1$.

Step 4

Make the values of A , a_0, \dots, a_{k-1} and C_1, \dots, C_{k-1} public.

To verify *Alice*'s identity, *Alice* and *Bob* use the following procedure:

Verification Protocol

Step 1

Bob selects at random an s -sparse essentially $\{0, 1\}$ -polynomial $h(X) \in \mathbb{F}_q[X]$ with $h(0) = B$ and sends it to *Alice*.

Step 2

Alice selects at random an r -sparse $\{0, 1\}$ -polynomial $g(X) \in \mathbb{F}_q[X]$ of degree at most $q - 1$ with $g(0) = 1$.

Step 3

Alice computes

$$F(X) \equiv f(X)g(X)h(X) \pmod{X^q - X}$$

and

$$D_j = g(a_j), \quad j = 1, \dots, k - 1,$$

and sends the polynomial F and D_1, \dots, D_{k-1} to *Bob*.

Step 4

Bob computes

$$E_j = h(a_j), \quad j = 1, \dots, k - 1,$$

and verifies that $F(X)$ is an rst -sparse $\{0, 1, A, B, AB\}$ -polynomial with $F(0) = AB$, and

$$F(a_j) = C_j D_j E_j, \quad j = 0, \dots, k - 1,$$

where $D_0 = E_0 = 1$, $C_0 = 0$.

Of course, there is a negligible chance that the constructed polynomial $F(X)$ is not a $\{0, 1, A, B, AB\}$ -polynomial, however if rst is substantially smaller than q this chance is extremely small (and in this case *Alice* and *Bob* can always repeat the procedure).

The sparsity of the polynomials involved guarantees the computational efficiency of this scheme.

In particular, using repeated squaring one can compute any power a^e with $a \in \mathbb{F}_q$ and an integer e , $0 \leq e \leq q - 1$, in about $2 \log q$ arithmetic operations in \mathbb{F}_q in the worst case and about $1.5 \log q$ arithmetic operations on average; see Section 1.3 of [1], Section 4.3 of [2], or Section 2.1 of [3]. Thus any τ -sparse $G(X) \in \mathbb{F}_q[X]$ can be evaluated at any point in about $O(\tau \log q)$ arithmetic operations in \mathbb{F}_q .

It is also well known that any element of \mathbb{F}_q can be encoded by using about $\log q$ bits.

Finally, we remark that if $0 \in S \subseteq \mathbb{F}_q$ then any τ -sparse S -polynomial $G(X) \in \mathbb{F}_q[X]$ of degree at most $q - 1$ can be encoded with about $\tau \log(q|S| - q)$ bits. Indeed, we have to identify at most τ positions at which G has a non-zero coefficient. Encoding of each position requires about $\log q$ bits. For each such position, about $\log(|S| - 1)$ bits are then required to determine the corresponding element of S .

For example, the signature must encode rst positions of the polynomial F (corresponding to its non-zero coefficients), which takes about $rst \log q$ bits. Each position requires two additional bits to distinguish between the possible coefficients 1, A , B and AB . The encoding of D_1, \dots, D_{k-1} requires about $(k-1) \log q$ bits. Thus, the total signature size is $(rst + k - 1) \log q + 2rst$ bits.

Putting everything together, after simple calculations we derive that, using the naive repeated squaring exponentiation,

- the *initial set-up* takes $O(kt \log q)$ arithmetic operations in \mathbb{F}_q ;
- the *private key size* is about $(t+1) \log q$ bits;
- the *public key size* is about $2k \log q$ bits;
- *signature generation*, that is, computation of the polynomial F and elements D_j , $j = 0, \dots, k-1$, takes $O(rst)$ arithmetic operations with integers in the range $[0, 2q-2]$ and $O((k-1)r \log q)$ arithmetic operations in \mathbb{F}_q ;
- the *signature size* is about $(rst + k - 1) \log q + 2rst$ bits;
- *signature verification*, that is, computation $F(a_j)$ and the products $C_j D_j E_j$, $j = 0, \dots, k-1$, takes about $O(ksrt \log q)$ arithmetic operations in \mathbb{F}_q .

We remark that the practical and asymptotic performance of this scheme can be improved if one uses more sophisticated algorithms to evaluate powers and sparse polynomials; see [1, 2, 3, 9, 11]. In particular, one can use precomputation of certain powers of the a_j 's and several other clever tricks which we do not consider in this paper.

3 Possible Attacks

It is clear that recovering or faking the private key (that is, finding a t -sparse essentially $\{0, 1\}$ -polynomial polynomial $\tilde{f}(X) \in \mathbb{F}_q[X]$ with $\tilde{f}(0) = A$, and $\tilde{f}(a_j) = C_j$, $j = 0, \dots, k-1$, $C_0 = 0$) or faking the signature (that is, finding a rst -sparse $\{0, 1, A, B, AB\}$ -polynomial $\tilde{F}(X) \in \mathbb{F}_q[X]$ with $\tilde{F}(0) = AB$, and $\tilde{F}(a_j) = C_j D_j E_j$, $j = 0, \dots, k-1$) represent the same problem (with slightly different parameters).

We also remark that that without the reduction

$$f(X)g(X)h(X) \pmod{X^q - X},$$

one of the one possible attacks would be via polynomial factorization. In particular, in a practical implementation of this scheme, one should make sure that both f and g have terms of degree greater than $q/2$ (so there are some reductions). Moreover, even without the reduction modulo $X^q - X$, the factorization attack does not seem to be feasible because of the large degrees of the polynomials involved; all known factorization algorithms (as well as their important components such as irreducibility testing and the greatest common divisor algorithms)

do not take advantage of sparsity or any special structure of the coefficients; see [2, 10]. In fact the first factor any of this algorithms will find will be the trivial one, that is, $X - a_0$. However the quotient $F(X)/(X - a_0)$ is most likely neither sparse nor a $\{0, 1\}$ -polynomial.

It is also possible that by using some “clever” choice of polynomials h , after several rounds of identification, *Bob* will be able to gain some information about f . Although the polynomials g are supposed to prevent him from doing this, in the next section we present another idea, which can be applied to other situations and which should make this attack completely unfeasible.

One might also consider lattice attacks. In the abstract, they could succeed, but since the dimension of the lattice would be equal to the (very large) degree of the polynomials involved, any such attack would be completely unfeasible. In particular, with current technology one can reduce lattices of degrees in the hundreds, while our lattices will have dimension roughly 2^{31} .

Finally, the probability of success in a brute force attack to discover or fake the signature, when the attacker selects a random rst -sparse $\{0, 1, A, B, AB\}$ -polynomial $\tilde{F}(X) \in \mathbb{F}_q[X]$ with $\tilde{F}(0) = AB$ that verifies only if $\tilde{F}(a_j) = C_j D_j E_j$, $j = 0, \dots, k-1$, is about

$$\min \left\{ 4^{-rst+1} \binom{q-1}{rst-1}^{-1}, q^{-k} \right\}.$$

Similarly, the probability of randomly guessing or faking the private key f is

$$\min \left\{ \binom{q-1}{t-1}^{-1}, q^{-k} \right\}.$$

In particular, we do not see any security flaws in this scheme even if $k = 1$. While the choice $k = 1$ has the obvious advantage of minimizing the signature size (in this case, *Alice* sends only the polynomial F), in order to provide the required level of security, quite large values of q must be used. From a practical standpoint, it is very convenient to work in the field with $p = 2^{31} - 1$ elements. Thus, to guarantee the 2^{90} level of security, which is currently accepted as standard, it is better to take $k \geq 3$. We believe that the choices $q = p = 2^{31} - 1$, $r = s = t = 5$, and $k = 3$ provide a fast, short (about 4200 bits long), and reliable signature.

4 Modification of the Basic Scheme

To prevent *Bob* from gaining any useful information about f by selecting some special polynomials h , *Alice* can select $a \in \mathbb{F}_q$ and two t -sparse essentially $\{0, 1\}$ -polynomials $f_1(X), f_2(X) \in \mathbb{F}_q[X]$ of degree at most $q-1$ which for some $A, C_1, \dots, C_{k-1} \in \mathbb{F}_q$ and distinct $a_0, \dots, a_{k-1} \in \mathbb{F}_q$ satisfy the conditions

$$f_1(0) = f_2(0) = A \tag{1}$$

and

$$f_1(a_j) = f_2(a_j) = C_j, \quad j = 0, \dots, k-1, \quad (2)$$

where $C_0 = 0$.

To do so, *Alice* selects a certain parameter $n < t$, considers random $\{-1, 1\}$ -polynomials $\psi(X)$ and tries to find a root of this polynomial over \mathbb{F}_q . For values of n of reasonable size this can be done quite efficiently, at least in probabilistic polynomial time; see [2,10].

It follows from Theorem 3 of [8] that for sufficiently large q the probability of a monic polynomial of degree n over \mathbb{F}_q having k distinct roots in \mathbb{F}_q is

$$P_k(n, q) = \sum_{m=k}^{\infty} \binom{q}{m} q^{-m} \sum_{l=0}^{n-m} (-1)^l \binom{q-m}{l} q^{-l}.$$

In particular,

$$\lim_{n \rightarrow \infty} \lim_{q \rightarrow \infty} P_k(n, q) = \frac{1}{k! e^k}.$$

Therefore, after $O(k! e^k)$ *Alice* will find with high probability an n -sparse $\{-1, 1\}$ -polynomial $\psi(X) \in \mathbb{F}_q[X]$, having k distinct roots $a_0, \dots, a_{k-1} \in \mathbb{F}_q$. Then she can write $X\psi(X) = \varphi_1(X) - \varphi_2(X)$ where φ_1, φ_2 are $\{0, 1\}$ -polynomials. Obviously,

$$\varphi_1(a_j) = \varphi_2(a_j), \quad j = 0, \dots, k-1,$$

and

$$\varphi_1(0) = \varphi_2(0) = 0$$

Then *Alice* selects a random $(t-n)$ -sparse $\{0, 1\}$ -polynomial $\varphi(X) \in \mathbb{F}_q[X]$ of degree at most $q-1$ and with $\varphi(0) = 0$. Now *Alice* puts

$$f_i(X) = \varphi(X) + \varphi_i(X) + A, \quad i = 1, 2,$$

where

$$A = -\varphi(a_0) - \varphi_1(a_0) = -\varphi(a_0) - \varphi_2(a_0).$$

Thus f_1 and f_2 are t -sparse essentially $\{0, 1\}$ -polynomials which satisfy (1) and (2). Therefore, now *Alice* can alternate f_1 and f_2 in a random order.

Instead of the sum $\varphi(X) + \varphi_i(X)$, $i = 1, 2$, one can also consider more complicated linear combinations with $\{0, 1\}$ -polynomial coefficients. For example, one can put

$$f_i(X) = \varphi(X) + \psi(X)\varphi_i(X) + A, \quad i = 1, 2,$$

for a random $\{0, 1\}$ -polynomial $\psi(X) \in \mathbb{F}_q[X]$ and

$$A = -\varphi(a_0) - \psi(a_0)\varphi_1(a_0) = -\varphi(a_0) - \psi(a_0)\varphi_2(a_0).$$

5 Concluding Remarks

It is natural to try to construct and use more than two t -sparse essentially $\{0, 1\}$ -polynomials which take the same value at k distinct points. However our approach of Section 4 does not seem to extend to this case.

Acknowledgments. A part of this work was done during visits by W. B. and D.L. to Macquarie University, whose hospitality and support are gratefully acknowledged.

Work supported in part, for D. L. by the National Science Foundation and a Big 12 Faculty Fellowship from the University of Missouri and for I. S. by the Australian Research Council.

References

1. H. Cohen *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1997. [70](#), [71](#)
2. J. von zur Gathen and J. Gerhard, *Modern computer algebra*, Cambridge Univ. Press, Cambridge, 1999. [70](#), [71](#), [72](#), [73](#)
3. D. M. Gordon, ‘A survey of fast exponentiation methods’, *J. Algorithms*, **27** (1998), 129–146. [70](#), [71](#)
4. D. Grant, K. Krastev, D. Lieman and I. E. Shparlinski, ‘A public key cryptosystem based on sparse polynomials’, In: *Proc. International Conference on Coding Theory, Cryptography and Related Areas, Guanajuato, 1998, Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, 1999 (to appear). [68](#)
5. J. Hoffstein, B. S. Kaliski, D. Lieman, M. J. B. Robshaw and Y. L. Yin, ‘A new identification scheme based on polynomial evaluation’, *Preprint*, 1997, 1–9. [68](#)
6. J. Hoffstein, D. Lieman and J. H. Silverman, ‘Polynomial Rings and Efficient Public Key Authentication’, In: *Proc. the International Workshop on Cryptographic Techniques and E-Commerce*, City University of Hong Kong Press, to appear. [68](#)
7. J. Hoffstein, J. Pipher and J. H. Silverman, ‘NTRU: A ring based public key cryptosystem’, *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1433** (1998), 267–288. [68](#)
8. A. Knopfmacher and J. Knopfmacher, ‘Counting polynomials with a given number of zeros in a finite field’, *Linear and Multilinear Algebra*, **26** (1990), 287–292. [73](#)
9. N. Pippenger, ‘On the evaluation of powers and monomials’, *SIAM J. Comp.*, **9** (1980), 230–250. [71](#)
10. I. E. Shparlinski, *Finite fields: Theory and computation*, Kluwer Acad. Publ., Dordrecht, 1999. [72](#), [73](#)
11. A. C.-C. Yao, ‘On the evaluation of powers’, *SIAM J. Comp.*, **5** (1976), 100–103. [71](#)

A State-Based Model for Certificate Management Systems

Chuchang Liu, Maris A. Ozols, Marie Henderson, and Tony Cant

Information Technology Division
Defence Science and Technology Organisation
PO Box 1500, Salisbury
SA 5108, Australia

{Chuchang.Liu,Maris.Ozols,Marie.Henderson,Tony.Cant}@dsto.defence.gov.au

Abstract. A Certificate Management System (CMS) is used to generate, distribute, store and verify certificates. It supports secure electronic communication through its functions. This paper presents a state-based model for certificate management systems. The axiomatization of CMS structures and the security policy followed by CMSs is discussed. The main functions of a CMS, including certificate issuing, certificate revocation and certificate rekeying, are formally described through transitions that change states of the CMS. A major CMS client function, certificate verification, is also formally discussed. With this model, an approach to the formal specification of the structure and behavior of a CMS is provided. The approach is very general, and would be useful in guiding the developer and the evaluator of a CMS with the design, analysis and implementation of the system.

Keywords: certificate, CA (Certification Authority), certificate management systems, certificate verification, information security, formal methods.

1 Introduction

Certificates have been used with security applications to support the validation of digital signatures on documents that are exchanged through computer networks. Recently, there has been substantial interest in public key technologies [2,3,4] being used to support secure electronic communications. In order to use public key cryptography, it is necessary to make an entity's public key available to others in such a way that its authenticity (i.e., its status as the true public key of that entity) and validity are verifiable. Public key certificates can be viewed as a vehicle by which public keys may be stored, distributed or forwarded over unsecured media while minimising the danger of undetectable manipulation. Anyone who wants to use a certificate must have a valid copy of the public key of the Certification Authority (CA) who issued the certificate, and must trust the CA.

A Certificate Management System (CMS) can provide mechanisms for managing public key certificates through its administration functions. It is used to

generate, distribute, store and verify certificates. Practical discussions on the structure of a CMS, its establishment, as well as functions and protocols between its components can be found in a number of research papers and references, such as in Kapidzic [6,7], Trcek [11], and PKIX Working Group Internet Draft [5].

As is well known, formal methods can provide strict proof technologies for verifying critical properties of a security system in a precise and unambiguous way, and also guide the developer towards a design of the security architecture of the system and its implementation. Formal techniques have been used for specifying authorization requirements [12] and analysing security properties of cryptographic protocols [10]. Formal methods have also been applied in the design of security devices. However, there has been a distinct lack of formal methods and techniques which can be used for the description of the structure and the behavior of CMSs. Therefore, theoretical studies of CMSs that may lead to a formal specification for the structure, functions and security properties of such systems are highly desirable.

In this paper, we present a state-based model for CMSs. One of our contributions is to propose an approach to the formalization of CMS structures and the security policy. The axiomatization of the topology structure and states of a CMS is discussed, and a general security policy followed by CMSs is also formalized. The other contribution is to provide a formal description for the main functions of a CMS, including certificate issuing, certificate revocation and certificate rekeying. All these function are described as transitions that change states of the CMS. A major CMS client function, certificate verification, is also formally discussed.

Our discussion is based on certificate management systems with top-down hierarchical structure, but the method is very general. Therefore, the methods and techniques proposed in this paper are suitable for any other kind of certificate management systems, and they would be useful in guiding the developer and the evaluator of a CMS with the design, analysis and implementation of the system.

The rest of the paper is structured as follows. Section 2 introduces the state-based model for CMSs, and gives a formal definition. Section 3 discusses the axiomatization of the CMS security policy, including certificate issuing policy, access control policy and trust policy. Section 4 formalizes the main CMS functions – certificate issuing, certificate revocation, and certificate rekeying – by defining appropriate transitions. The certificate verification function is discussed in Section 5, where a number of verification rules are presented. The last section concludes this paper with a short discussion about future work.

2 The Model

In this section, we present the state-based model for certificate management systems. An axiomatization of the CMS structure is discussed, and a formal definition of a CMS is given.

2.1 Certification Topology

We classify the entities or agents in a CMS into two classes, *Certification Authorities* (CAs) and *users*. CAs can have their own certificates, and they also issue certificates for others within the CMS. Users, also called *end entities*, are people or devices that may hold certificates issued by some CAs, but cannot issue any certificates themselves.

A user of a security service requiring knowledge of a public key generally needs to obtain and validate the certificate containing the required public key. If the user does not already hold an assured copy of the public key of the CA who signed the certificate, then he might need an additional certificate to obtain that public key. In general, a chain of multiple certificates may be needed, comprising the certificate of the public key owner signed by one CA, and zero or more additional certificates of CAs issued by other CAs. Such chains of certificates are called *certificate paths*.

There are different ways in which CMS entities might be configured so that public key users are able to find certificate paths. For instance, RFC 1422 [8] defines a rigid hierarchical structure for the Internet Privacy Enhanced Mail (PEM) [5], in which there are three types of PEM CAs: Internet Policy Registration Authority (IPRA) acts as the root of the PEM certification hierarchy at level 0, and issues certificates only for the next level of authorities, called PCAs (Policy Certification Authorities); PCAs, at level 1 of the PEM certification hierarchy, take the responsibility for establishing and publishing the certification policy with respect certifying users or subordinate certification authorities; and CAs, which are at level 2 of the hierarchy and can also be at lower levels (those at level 2 are certified by PCAs).

With our intention being to discuss CMSs in general, we assume that all the entities of a CMS, we call them agents, form a single root top-down certification hierarchy. This means that there is a special CA at the root, called the PAA (Policy Approval Authority), whose role is different from that of the other CAs. The PAA can certify others (but no one certifies it) and it holds a certificate which is trusted by everyone. All CAs can only issue certificates to their children, and no certificates are issued by users. In our model, the top-down certification hierarchy is formally described using a series of agent axioms.

Definition 1 Let Ω be a set of agents (including a special agent, called the PAA) and \downarrow a binary relation over Ω . Then $\langle \Omega, \downarrow \rangle$ is called a certification topology if it satisfies the agent axioms AA1 – AA6 given below.

Agent Axioms

- (AA1) $\forall X \bullet (\text{IsCA}(X) \vee \text{IsUsr}(X))$
- (AA2) $\forall X \bullet (\text{IsCA}(X) \leftrightarrow \exists Y \bullet (X \downarrow Y))$
- (AA3) $\forall X \bullet (\text{IsUsr}(X) \leftrightarrow \forall Y \bullet \neg(X \downarrow Y))$
- (AA4) $\forall X \bullet (X = \text{PAA} \leftrightarrow (\text{IsCA}(X) \wedge \neg \exists Y \bullet (Y \downarrow X)))$
- (AA5) $\forall X \bullet \neg(X \downarrow X)$
- (AA6) $\forall X \bullet \forall Y \bullet (X \downarrow^+ Y \rightarrow \neg(Y \downarrow^+ X))$

Here, $\text{IsCA}(X)$ means X is a CA, and $\text{IsUsr}(X)$ means X is a user. The axiom AA6 includes a new relation operator \downarrow^+ as a transitive closure of the relation \downarrow , which is defined by the following formula:

$$\forall X \bullet \forall Y \bullet (X \downarrow^+ Y \leftrightarrow (X \downarrow Y) \vee \exists Z \bullet (X \downarrow^+ Z \wedge (Z \downarrow^+ Y)))$$

Note that axioms AA2 and AA4 together have ruled out the trivial case that Ω has only one agent, the PAA. Therefore, in this structure, the PAA is always a CA.

A certification topology has an intuitive direct graphical representation – all agents (i.e., CAs and users) are represented as nodes and, for any $X, Y \in \Omega$, there is a directed edge from X to Y if and only if $X \downarrow Y$ holds (in the case, we say X is a *parent* of Y or Y is a *child* of X). The PAA as a special agent at the root has no parents. An example of a certificate topology is shown in Figure 1.

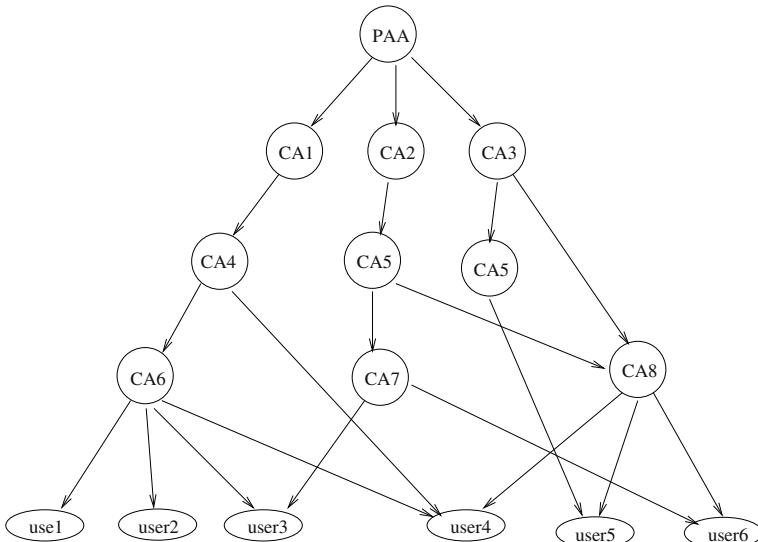


Fig. 1. A certification topology

The topology gives a trust model for certificate management. A search for finding a certificate path, as we will see, can be done based on the certification topology. In fact, certificate paths correspond to directed paths in the graphical representation of the topology. Based on the topology axioms, it is not difficult to show that, for any CA or any user, there exists at least one directed path to it from the PAA.

2.2 Certificates

A public-key certificate is a data structure consisting of a *data* part and a *signature* part. The data part contains, as a minimum, a public key and a string identifying the party (*subject*) to be associated therewith and the *issuer* of the certificate. The signature part consists of the digital signature of the issuer over the data part, thereby binding the subject's identity to the specified public key.

The certificates we consider have a standard public-key certificate format with the basic information outlined below.

Definition 2 A certificate has the following form:

$$\text{Cert } (\mathbf{I}, D_s, D_e, S, \text{PK}_S, \text{SIG}_{\mathbf{I}}(\mathbf{I}, D_s, D_e, S, \text{PK}_S))$$

or, simply, written as $\text{Cert } (\mathbf{I}, D_s, D_e, S, \text{PK}_S, \text{SIG}_{\mathbf{I}})$, where \mathbf{I} is the issuer, D_s and D_e are the start date and expiry date respectively, S is the subject of the certificate, PK_S is the public key of S , and $\text{SIG}_{\mathbf{I}}$ is the signature of the issuer \mathbf{I} .

There is a special certificate, named as **PAAcert**, that has the special form as follows:

$$\text{Cert } (-, -, -, \text{PAA}, \text{PK}_{\text{PAA}}, -)$$

PAAcert is intended to be the certificate of the PAA. The PAA is the top CA: its certificate is not issued by any other CA, so **PAAcert** is usually viewed as a self-signed certificate. In the case, the issuer's name, the start date, the expiry date and the signature parts are all omitted. All CAs and users would trust this certificate held by the PAA.

We denote the set of all certificates issued by CAs in the CMS as \mathcal{C} , called the *total certificate set* of the CMS. This means that any certificate we consider at any time must belong to \mathcal{C} . For any certificate $C \in \mathcal{C}$, the following projection functions can be used to obtain the value of each component contained in C . Let $C = \text{Cert } (\mathbf{I}, D_s, D_e, S, \text{PK}_S, \text{SIG}_{\mathbf{I}}(\mathbf{I}, D_s, D_e, S, \text{PK}_S))$, we define:

$$\begin{array}{ll} \overline{\mathbf{I}}(C) = \mathbf{I} & \overline{D_s}(C) = D_s \\ \overline{D_e}(C) = D_e & \overline{S}(C) = S \\ \overline{\text{PK}}(C) = \text{PK}_S & \overline{\text{SIG}}(C) = \text{SIG}_{\mathbf{I}}(\mathbf{I}, D_s, D_e, S, \text{PK}_S) \end{array}$$

These functions will be used frequently in our discussion.

2.3 CMS States and Transitions

At any moment in time, or at any given state, an agent in the CMS should hold zero or more certificates and, in particular, any CA should also be associated with a CRL (Certificate Revocation List) issued by itself periodically. Thus, we define states of a CMS as follows:

Definition 3 Let $\langle \Omega, \downarrow \rangle$ be a certification topology and \mathcal{C} the total certificate set. A relation s from Ω to $2^{\mathcal{C}} \times 2^{\mathcal{C}}$ is called a CMS state if it satisfies the state axioms SA1 – SA3 given below, where $2^{\mathcal{C}}$ is the power set of \mathcal{C} . Under a state s , we call $s(\mathbf{X}, \zeta, \eta)$ a triple, where $\zeta (\subset \mathcal{C})$ is a set of certificates issued to \mathbf{X} and $\eta (\subset \mathcal{C})$ is a set of certificates issued by \mathbf{X} .

State Axioms

- (SA1) $\forall \mathbf{X} \bullet \exists \zeta \bullet \exists \eta \bullet (s(\mathbf{X}, \zeta, \eta) \wedge \forall \zeta' \bullet \forall \eta' \bullet (s(\mathbf{X}, \zeta', \eta') \rightarrow (\zeta' = \zeta) \wedge (\eta' = \eta)))$
- (SA2) $\forall \mathbf{X} \bullet \forall \mathbf{C} \bullet (s(\mathbf{X}, \zeta, \eta) \wedge \mathbf{C} \in \zeta \rightarrow \overline{S}(\mathbf{C}) = \mathbf{X})$
- (SA3) $\forall \mathbf{X} \bullet \forall \mathbf{C} \bullet (s(\mathbf{X}, \zeta, \eta) \wedge \mathbf{C} \in \eta \rightarrow \overline{I}(\mathbf{C}) = \mathbf{X})$

According to Definition 3, at any given state s (referred to as the current state), any agent \mathbf{X} is associated with two certificate sets: ζ and η . All certificates in ζ should be possessed by \mathbf{X} because their subject is \mathbf{X} ; and all certificates in η must be issued by \mathbf{X} itself at sometime because their issuer is \mathbf{X} . Thus, ζ may be viewed as the possessed certificate set of \mathbf{X} , while η may be viewed as the revoked certificate set of \mathbf{X} . Formally, we have

Definition 4 Let s be a CMS state. If we have $s(\mathbf{X}, \zeta, \eta)$, then ζ is called the possessed certificate set of \mathbf{X} , which lists all certificates possessed by \mathbf{X} , and η is called the revoked certificate set of \mathbf{X} , which represents the CRL issued by \mathbf{X} at the state s . In particular, if $\mathbf{C} \in \zeta$, we say that \mathbf{X} holds the certificate \mathbf{C} ; if $\mathbf{C} \in \eta$, we say that \mathbf{X} has revoked the certificate \mathbf{C} .

In the following, for the reason of simplifying our discussion, we will often use $\zeta_{\mathbf{X}}$ and $\eta_{\mathbf{X}}$ to denote the possessed certificate set and the revoked certificate set of an agent \mathbf{X} , respectively, at a given CMS state.

In the theory of state machines, transitions are usually described as actions that change states of a machine. Adopting the formal approach to defining a state machine proposed by Eastaughffe *et al.* [1], we define transitions of the CMS as follows:

Definition 5 A CMS transition changes one CMS state into another and consists of three parts:

- the **Let declaration**, which introduces local variables for abbreviating expressions within the transition;
- the **guard**, denoted as **Pre**, a boolean expression which must be true for the current values of the variables; and
- the **action list Act**, essentially a parallel assignment statements, which involve the state changes.

As an example, a simple transition is given as follows. Suppose $\mathbf{Y} \downarrow \mathbf{X}$, and, at the current state s , we have $s(\mathbf{X}, \zeta_{\mathbf{X}}, \eta_{\mathbf{X}})$ and now a transition involving only the issue by \mathbf{Y} of a new certificate \mathbf{C} to \mathbf{X} is applied, then the transition might be defined as:

Let: $C = \text{Cert}(Y, D_s, D_e, X, \text{PK}_X, \text{SIG}_Y)$
 Pre: $\text{SucReqCert}(X, Y)$
 Act: $\zeta_X \leftarrow \zeta_X \cup \{C\}$

This definition means that, if the checking of the certificate request “ X requests a certificate issued by Y ” is successful, Y will issue a certificate to X . Therefore, when $\text{SucReqCert}(X, Y)$ has the value “True”, at the next state, denoted as s' , the set of certificates held by X will be $\zeta_X \cup \{C\}$. Thus, we have $s'(X, \zeta_X \cup \{C\}, \eta)$. This transition does not involve any other actions than issuing the certificate C to X by Y . Therefore, for any $Z \in \Omega$, if $s(Z, \zeta_Z, \eta_Z)$ and $Z \neq X$, then $s'(Z, \zeta_Z, \eta_Z)$.

An expanded discussion on transitions will be given in Section 4.

2.4 A Formal Definition of a CMS

We are now able to give the formal definition of a CMS as follows.

Definition 6 A CMS (Certificate Management System) is defined as a 5-tuple $\langle \Omega, \downarrow, \mathcal{C}, \mathcal{S}, \mathcal{T} \rangle$ where

- Ω is a set of agents, including the special agent PAA, and Ω with the relation \downarrow forms a certification topology $\langle \Omega, \downarrow \rangle$;
- \mathcal{C} is a non-empty set of certificates, called the total certificate set, that contains all possible certificates issued by CAs and a special element called the PAAcert, which is held by the PAA.
- \mathcal{S} is a set of CMS states; and
- \mathcal{T} is a set of transitions, a transition may change one state into another when it is applied.

According to the definition, a CMS must satisfy all the axioms AA1 – AA6 and SA1 – SA3.

3 The CMS Policy

In this section, we present a general security policy for CMSs, which involves certificate issuing, access control, the validity of certificates, and the distribution of trust.

We assume that the operations that can be taken by agents are classified into two types: CMSservice and other. To formalize the general security policy, we introduce several predicates as follows:

- | | |
|-----------------------------|---|
| $\text{CanCertify}(X, Y)$: | X can certify Y . |
| $\text{ActType}(a, Ty)$: | The type of an operation a is Ty . |
| $\text{CanTake}(X, a)$: | X can perform the operation a . |
| $\text{Valid}(X)$: | X is valid, where X can be a certificate, a public key, or a signature. |
| $\text{Holds}(X, C)$: | X holds the certificate C . |
| $\text{HasRevoked}(X, C)$: | X has revoked the certificate C . |
| $\text{Trusts}(X, C)$: | X trusts the certificate C . |

3.1 Certificate Issuing Policy

Let $\langle \Omega, \downarrow, \mathcal{C}, \mathcal{S}, \mathcal{T} \rangle$ be a CMS. Then, by the agent axioms, we can directly obtain the following conclusion: For any $X \in \Omega$, its certification domain is defined. Formally, we define

$$\forall X \bullet (\text{IsCertDom}(\mathcal{D}, X) \leftrightarrow \forall Y \bullet (Y \in \mathcal{D} \leftrightarrow X \downarrow Y))$$

where $\text{IsCertDom}(\mathcal{D}, X)$ means that \mathcal{D} is the certification domain of X .

This definition does not require that X must be a CA; it can actually be any agent in the CMS. However, if X is a user, using the axiom AA3, it is easy to show that \mathcal{D} is empty. That is, the certification domain of any user is the empty set.

Thus, the certificate issuing policy can simply be stated as follows:

- In a CMS, an agent can certify only those who belong to its certification domain.

This policy can be formalized as the following axiom:

$$(PA1) \quad \forall X \bullet \forall Y \bullet (\text{CanCertify}(X, Y) \rightarrow Y \in \mathcal{D} \wedge \text{IsCertDom}(\mathcal{D}, X))$$

An obvious corollary of the axiom PA1 is that users cannot certify anybody, because their certification domains are empty.

As an alternative of PA1, we may have

$$(PA1') \quad \forall X \bullet \forall Y \bullet (\text{CanCertify}(X, Y) \rightarrow X \downarrow Y)$$

It is easy to show that PA1 and PA1' are logically equivalent. However, we prefer PA1 due to a technical reason: the amount of work in checking if someone is an eligible member to whom an entity can issue a certificate based on the the certificate domain of this entity is usually less than that based on the relation \downarrow if the certificate domain has already been calculated.

3.2 Access Control Policy

If a user requests a CMS service, he may be able to provide a valid certificate; but in some case he may not have one. Also attackers may try to use forged or stolen certificates to access the system. In the state-based model, to exclude unauthorised users, the CMS follows a basic rule – the access control policy stated as follows.

- No one without a valid certificate can be allowed to use security applications (CMS services).

This policy is formalized as the axiom:

$$(PA2) \quad \forall X \bullet \forall a \bullet (\text{CanTake}(X, a) \rightarrow \text{ActType}(a, \text{other}) \vee \exists C \bullet (\text{Holds}(X, C) \wedge \text{Valid}(C)))$$

This says: for any agent X and any operation a , if X can perform the operation a , then the type of a is **other** or X holds a valid certificate.

3.3 The Validity of Certificates

According to the access control policy, any user who is applying for CMS services should be able to provide a valid certificate issued by a CA. In other words, the CMS must verify the validity of every certificate that a security application uses. Therefore, the CMS should provide support for the verification process to check if a certificate is valid. The CMS policy regarding the validity of certificates include the following items:

- The certificate **PAAcert** is held by the PAA and it is valid.
- A certificate held by someone other than the PAA is valid if and only if the following conditions are all satisfied: (1) the issuer of the certificate is a parent of this person, (2) the current time belongs to the time interval shown on the certificate, (3) the signature of the issuer is valid, and (4) the certificate has not been revoked by the issuer.
- The signature on a certificate is valid if and only if the issuer of the certificate holds a valid certificate.

In the state-based model, all these items have been formalized as the following axioms:

- $$\begin{aligned}
 (\text{PA3}) \quad & \text{Holds(PAA, PAAcert)} \wedge \text{Valid(PAAcert)} \\
 (\text{PA4}) \quad & \forall X \bullet \forall C \bullet (\text{Holds}(X, C) \wedge \neg(X = \text{PAA}) \rightarrow (\text{Valid}(C) \leftrightarrow \overline{I}(C) \downarrow X \wedge \\
 & (\text{D}_s(C) \leq \text{Today} \leq \text{D}_e(C)) \wedge \text{Valid}(\overline{\text{SIG}}(C)) \wedge \neg\text{HasRevoked}(\overline{I}(C), C))) \\
 (\text{PA5}) \quad & \forall C \bullet (\text{Valid}(\overline{\text{SIG}}(C)) \leftrightarrow \exists C' \bullet (\text{Holds}(\overline{I}(C), C') \wedge \text{valid}(C')))
 \end{aligned}$$

where **Today** is variable representing the current time.

3.4 Trust Policy Axioms

From the above policy axioms, we can see that in general a single certificate is not enough to establish assurance that the certificate subject is authorised to access a CMS service it is requesting. Usually, a certificate path that starts from the certificate held by the trusted issuer is needed.

We make the following assumptions concerning trust within the CMS:

- (1) CAs and users trust all CAs to faithfully execute their CA operations; and
- (2) CAs and users trust that it is not viable to tamper with CMS certificates.

These assumptions can be well founded and supported by CMS practices. Firstly, assurance is provided for (1) through the use of accreditation of CAs, Certificate Practice Statements published by CAs and the implementation of appropriate policy¹. Assurance is provided for (2) through the use of digital signatures, and good control of private keys.

¹ Note that policy for CAs can be listed and checked in much the same way as in which certificates are checked and can even be included as an extension in certificates.

The only assumption made concerning trusted certificates is that the certificate **PAAcert**, held by the PAA, is trusted by all CAs and users within the CMS. In fact, since the CMS is a top-down hierarchy, trust of all CMS certificates can be determined using the PAA certificate and by checking the validity of certificates along the certification path. For example, suppose that a CA's certificate is valid and its issuer is the PAA whose certificate is trusted then, from our assumptions, the CA's certificate is trusted. It is seen that for the CMS trust must always be referenced back to the single trust point, namely, the PAA's certificate (as is the case in top-down hierarchies).

Trust is transferred throughout the CMS using the trust policy stated as follows:

- Everybody trusts the certificate **PAAcert**.
- One trusts a certificate other than the **PAAcert** if and only if the certificate is valid and the certificate's issuer holds a certificate that the person trusts.

We formalize the trust policy with the following axioms:

$$\begin{aligned} (\text{PA7}) \quad & \forall X \bullet \text{Trusts}(X, \text{PAAcert}) \\ (\text{PA8}) \quad & \forall X \bullet \forall C \bullet (\neg(C = \text{PAAcert}) \rightarrow (\text{Trusts}(X, C) \leftrightarrow \\ & \text{Valid}(C) \wedge \exists C' \bullet (\text{Holds}(\overline{I}(C), C') \wedge \text{Trusts}(X, C')))) \end{aligned}$$

In our model, the trust policy of the CMS consists of only the two axioms as given above. These axioms restrict the CMS to have only one trust point from which all trust is inferred. There are other ways to distribute trust within a CMS, such as multiple trust points, but these will not be considered here.

4 The Formalization of CMS Functions

In this section, we consider three CMS functions: certificate issuing, certificate revocation and certificate rekeying. All these functions of a CMS may lead to dynamic changes of state. We have already stated that the actions which change states of a CMS can be viewed as transitions within the system. Therefore, all these functions can formally be described through transitions.

4.1 Transitions

Let $\langle \Omega, \downarrow, \mathcal{C}, \mathcal{S}, T \rangle$ be a CMS. We now discuss transitions at a given state s , where $s \in \mathcal{S}$.

A simple transition is defined as one of the following actions:

- Issuing of a certificate C to an agent X by a CA Y ;
- Revoking of a certificate C by an agent X .
- Rekeying of a certificate C held by an agent X .

We denote the simple transitions involving the above actions as $\text{Issues}(Y, C, X)$, $\text{Revokes}(X, C)$, and $\text{Rekeys}(X, C)$ respectively.

Without loss of generalisation, a transition of a CMS may be a simple transition as above, or a compound transition formed from several simple transitions that occur in parallel at the same time. That is, if t_1, \dots, t_n are simple transitions, $t_1 \parallel \dots \parallel t_n$ is a compound transition, where \parallel is called the parallel operator over transitions.

4.2 Certificate Issuing

CMSs support two types of certification requests: *self-registration* and *agent-registration*. In a self-registration request, an Organization Registration Authority (ORA) may provide a secret message to the prospective certificate holder. The entity itself generates its own key pair, forms a certificate request, signs it with the corresponding private key material, and includes authentication information based on the secret message provided by the ORA. The CA receives the request, and verifies the requester's identity through the authentication information. If accepted, the CA will generate a new certificate and send it to the certificate holder.

In an agent-registration request, the agent vouches for the prospective certificate holder's identity and the binding to the public key. When a certificate request comes from an accredited agent, the receiving CA will process the request and, if accepted, generate a new certificate and send it to the agent. The CA may also send the new certificate to the holder. The CA may reject the agent-generated certification request due to the following reasons: the authentication of the identity of the certificate's subject fails, or the agent does not possess a valid certificate. If the CA rejects the request, it will report the failure to the agent stating the reason of the failure.

In any case, the action “issuing a certificate to X by Y ” can happen only when the checking of certificate request is successful. We use the predicate $\text{SucReqCert}(X, Y)$ to denote that the checking of certificate request “ X requests a certificate issued by Y ” is successful. Then the transition $\text{Issues}(Y, C, X)$ has the form given in Section 2.3.

The *certificates issuing* function must follow the certificate issuing policy. Therefore, the action “issuing a certificate to X by Y ” can be performed only when the following conditions are satisfied:

- Y can certify X , i.e., $Y \downarrow X$; and
- Y holds a valid certificate.

Thus, the value of the guard $\text{SucReqCert}(X, Y)$ can be obtained by the formula:

$$\text{SucReqCert}(X, Y) \leftrightarrow (Y \downarrow X) \wedge \exists C' \bullet (\text{Holds}(Y, C') \wedge \text{Valid}(C'))$$

By axiom PA1, if the prospective certificate holder in a certification request does not belong to the certification domain of the issuer required, the request must not be accepted. Usually, when a CA receives a certification request, it will

first check if that the prospective certificate holder appears in its certification domain. If not, it may immediately reject the request.

In the case that a number of issuing certificate actions happen at the same time, a compound transition may be needed. The definition of a compound transition can simply be formed by conjunction of the formulas appearing in the corresponding parts of all simple transitions constructing the compound transition. For instance, suppose “issuing a certificate C_1 to X_1 by Y_1 ”, …, “issuing a certificate C_n to X_n by Y_n ” happen at the same time, then the certificate issuing behavior can be described as the compound transition

$$\text{Issues}(Y_1, C_1, X_1) \parallel \dots \parallel \text{Issues}(Y_n, C_n, X_n),$$

whose definition as follows can directly be obtained from the simple transitions $\text{Issues}(Y_1, C_1, X_1)$, …, and $\text{Issues}(Y_n, C_n, X_n)$:

$$\begin{aligned} \text{Let: } & C_1 = \text{Cert}(Y_1, D_{s_1}, D_{e_1}, X_1, PK_{X_1}, SIG_{Y_1}), \\ & \dots \\ & C_n = \text{Cert}(Y_n, D_{s_n}, D_{e_n}, X_n, PK_{X_n}, SIG_{Y_n}) \\ \text{Pre: } & \text{SucReqCert}(X_1, Y_1) \\ & \dots \\ & \text{SucReqCert}(X_n, Y_n) \\ \text{Act: } & \zeta_{X_1} \leftarrow \zeta_{X_1} \cup \{C_1\} \\ & \dots \\ & \zeta_{X_n} \leftarrow \zeta_{X_n} \cup \{C_n\} \end{aligned}$$

However, we should note that, if there is more than one CA issuing a certificate to the same agent, the actions involved in those issues could be combined to a single action. For instance, suppose a compound transition contains two simple transitions:

$$\text{Issues}(Y_1, C_1, X) \text{ and } \text{Issues}(Y_2, C_2, X),$$

where both

$$\begin{aligned} C_1 &= \text{Cert}(Y_1, D_{s_1}, D_{e_1}, X, PK_X, SIG_{Y_1}) \text{ and} \\ C_2 &= \text{Cert}(Y_2, D_{s_2}, D_{e_2}, X, PK'_X, SIG_{Y_2}) \end{aligned}$$

are issued to X . We have the action $\zeta_X \leftarrow \zeta_X \cup \{C_1\}$ in the simple transition $\text{Issues}(Y_1, C_1, X)$, and the action $\zeta_X \leftarrow \zeta_X \cup \{C_2\}$ in the simple transition $\text{Issues}(Y_2, C_2, X)$. In this case, the two actions should be combined to a single action $\zeta_X \leftarrow \zeta_X \cup \{C_1, C_2\}$.

4.3 Certificate Revocation

At the initial time when the CMS starts, we may assume that the sets of revoked certificates for all CAs (as well as all users) are empty. At any time after the initialization, every CA in the CMS must periodically issue a CRL, which contains all the certificates that it has revoked.

Certificates are revoked by a CA who issued them when the CA has reason to believe that they can no longer be trusted. At any time, when a CA, say X , issues a certificate C , X should of course trust C . That is, the formula $\text{Trusts}(X, C)$ automatically obtains the truth value “True” at the same time when C is issued, and this value will keep until C will not be trusted by X due to some reasons. Therefore, if a CA X revokes a certificate C , then preconditions should include: the issuer of C is X , and X does not trust C . We define

$$\text{ApvRevCert}(X, C) \leftrightarrow (\bar{I}(C) = X) \wedge \neg \text{Trusts}(X, C)$$

where $\text{ApvRevCert}(X, C)$ means that a need “ X revokes the certificate C ” is approved. Thus, “ X revokes C ” is the simple transition $\text{Revokes}(X, C)$, which can be defined as follows:

Let:

Pre: $\text{ApvRevCert}(X, C)$

Act: $\eta_X \leftarrow \eta_X \cup \{C\}$

A CA periodically issues a CRL, which may revoke a number of certificates. For instance, suppose that, at the current time, the certificates that X is revoking include C_1, \dots, C_n , then the revocation action can be represented as the compound transition $\text{Revokes}(X, C_1) \parallel \dots \parallel \text{Revokes}(X, C_n)$. In the same way we used for the representation of certificate issuing functions, we can easily obtain the definition of the compound transition as follows:

Let:

Pre: $\text{ApvRevCert}(X, C_1)$

.....

$\text{ApvRevCert}(X, C_n)$

Act: $\eta_X \leftarrow \eta_X \cup \{C_1, \dots, C_n\}$

The definition of a compound transition involving the case where several CAs revoke a number of certificates can also be obtained from the definitions of simple transitions contained in the compound transition.

4.4 Certificate Rekeying

At any time after the CMS is initialized, it is possible that some CA may need to change the key pair (the public key and the corresponding private key) involved in a certificate held by itself due to an increased risk of the current keys being compromised.

The process of changing the keys of a CA for a certificate, performed by the certificate rekeying function, is as follows. When a new key pair is generated for a certificate by the CA who wants to change its key pair, a new certificate is created and a certificate signature request is sent to the parent who signed the old certificate. If the certificate rekeying request is accepted, the parent signs the new certificate and returns it to the CA. Rekeying a certificate must affect the

certificate hierarchy, since some certificates held by the CA's children may have been signed with the old private key of the CA. Therefore, upon receiving the new certificate, the CA needs to re-sign all those certificates of its subordinates with the new private key, which were signed with that old private key.

The certificate rekeying function can also formally be described using transitions. To do this, we first introduce two new predicates as follows:

- AcpReKey(X, C):** The request “X wants to rekey the certificate C” is accepted.
SIGKEY(C₁, C₂): The certificate C₁ is signed with the private key for which the corresponding public key is issued in the certificate C₂.

We now assume that X changes the key pair for a certificate C held by itself, and C₁, ..., C_n are all the certificates signed with the private key for which the corresponding public key is issued in the certificate C. Then this action can be expressed as the simple transition **Rekeys(X, C)**, which has the form as follows:

Let: $C = \text{Cert} \{Y, D_s, D_e, X, \text{PK}_X, \text{SIG}_Y\}$
 $C' = \text{Cert} \{Y, \text{Today}, D'_e, X, \text{PK}_X, \text{SIG}'_Y\}$
 $C_1 = \text{Cert} \{X, D_{s_1}, D_{e_1}, X_1, \text{PK}_{X_1}, \text{SIG}_{X_1}\}$
 $C'_1 = \text{Cert} \{X, \text{Today}, D'_{e_1}, X_1, \text{PK}_{X_1}, \text{SIG}'_{X_1}\}$
 \dots
 $C_n = \text{Cert} \{X, D_{s_n}, D_{e_n}, X_n, \text{PK}_{X_n}, \text{SIG}_{X_n}\}$
 $C'_n = \text{Cert} \{X, \text{Today}, D'_{e_n}, X_n, \text{PK}_{X_n}, \text{SIG}'_{X_n}\}$

Pre: **AcpReKey(X, C)**
SIGKEY(C₁, C)
 \dots
SIGKEY(C_n, C)

Act: $\zeta_X \leftarrow (\zeta_X \setminus \{C\}) \cup \{C'\}$
 $\zeta_{X_1} \leftarrow (\zeta_{X_1} \setminus \{C_1\}) \cup \{C'_1\}$,
 \dots
 $\zeta_{X_n} \leftarrow (\zeta_{X_n} \setminus \{C_n\}) \cup \{C'_n\}$

where **Today** is a variable representing the current time, and $\mathcal{A} \setminus \mathcal{B}$ stands for the difference set of the sets \mathcal{A} and \mathcal{B} .

Note that, when the certificate C is rekeyed to become the new certificate C', the old certificate C is automatically revoked by its issuer at the same time. For simplifying our discussion, we omit such an action.

Similarly, we can consider those cases where a number of CAs change the key pairs for a number of certificates at the same time. We may also consider a compound transition consisting of different types of simple transitions. There should not be any difficulty obtaining the definition of a compound transition from the definitions of simple transitions.

5 Certificate Verification

In a CMS, the certificate verification function is a major CMS client function, which is responsible for verifying the validity of every certificate that a security

application uses. Certificate verification proves that no one has tampered with the contents of the certificate, i.e. the public key, the validity date, and the identity of the certificate issuer and owner have not been changed. This relies on the properties of the digital signature². This section discusses the certificate verification principle, and presents verification rules.

5.1 The Certificate Verification Principle

Verifying a given certificate involves verifying the identity of the certificate issuer and owner, verifying the validity dates of the certificate, verifying the signature on the certificate, and verifying the certificate against the latest issuer's CRL list to make sure it has not been revoked. In order to verify the signature on the certificate, the certificate that the issuer holds should also be verified in the same way using the next higher certificate in the certificate path. Therefore, the verification process is iterative and terminates when a trusted certificate (by the verifier) is reached, or a invalid certificate is encountered.

When an invalid certificate is encountered, the certificate is not accepted as valid by the verifier. The verification process may begin by verifying the signature on the next certificate on the certification path with the PAA's public key from PAAcert. This process continues down the certification path until the required certificate is verified or an invalid certificate is encountered.

The certificate verification principle, which must be followed by all CMS users, can be stated as below:

- In a certificate verification process, when the verifier has found a certificate path constructed for verifying a required certificate in which all certificates are valid he may accept this certificate as valid, and in all other cases the certificate is regarded as invalid.

Note that, according to the certificate verification principle, it can happen that a certificate may actually be valid but the verifier did not find a corresponding certificate path in which all certificates are valid. In such a case, the verifier cannot accept this certificate as valid. This is the correct choice on security grounds.

5.2 Verification Rules

According to the above discussion, the process of obtaining and verifying the certificates from the trusted certificate to a required certificate is referred to as *certificate path development* and *certificate path verification*, respectively. An

² The digital signature provides a way to detect if any of these values have been changed. The original certificate issuer calculates a digital signature over the values that require protection and then issues the values and the signature which form the certificate. This signature can be checked by anyone who has a valid copy of the issuer's public key (obtained from the issuer's certificate), at a later stage. The details of the signature algorithm are beyond the scope of this paper.

efficient way to develop a certificate path is to start with the required certificate and build a certificate chain back towards a certificate trusted by the verifier. Certificate path verification can be done using certificate verification rules shown below. These rules are directly derived from the policy axioms.

A rule is usually expressed in the following form:

$$\frac{A_1 \quad A_2 \quad \dots \dots \quad A_n}{A}$$

which means that if formulas A_1, A_2, \dots, A_n are all proved to be true, then the fact “the formula A is true” is proved.

Verification Rules

$$(R1) \quad \frac{\overline{I}(C) \downarrow \overline{S}(C) \quad \overline{D}_s(C) \leq \text{Today} \leq \overline{D}_e(C) \quad \neg(C \in \eta_{\overline{I}(C)}) \quad \text{valid}(\overline{SIG}(C))}{\text{valid}(C)}$$

$$(R2) \quad \frac{C' \in \zeta_{\overline{I}(C)} \quad \text{valid}(C')}{\text{valid}(\overline{SIG}(C))}$$

$$(R3) \quad \frac{\neg(\overline{I}(C) \downarrow \overline{S}(C))}{\neg\text{Valid}(C)}$$

$$(R4) \quad \frac{(\text{Today} < \overline{D}_s(C)) \vee (\text{Today} > \overline{D}_e(C))}{\neg\text{Valid}(C)}$$

$$(R5) \quad \frac{C \in \eta_{\overline{I}(C)}}{\neg\text{Valid}(C)}$$

$$(R6) \quad \frac{\neg\text{Valid}(\overline{SIG}(C))}{\neg\text{Valid}(C)}$$

$$(R7) \quad \frac{\neg\exists C' \bullet (C' \in \eta_{\overline{I}(C)} \wedge \text{valid}(C'))}{\neg\text{valid}(\overline{SIG}(C))}$$

$$(RX) \quad \frac{\text{Trusts}(X, C)}{\text{Valid}(C)}$$

Rules (R1) and (R2) together form the basis for the verification of certificates; while rules (R3) – (R7) can be applied to promptly exclude unauthorised users by finding some flaw in their certificates. The rule (RX) depends on a particular verifier (X) and is related to trust.

Suppose a certificate path, say $\langle C_0, C_1, \dots, C_{n-1}, C_n \rangle$ has been developed for verifying the certificate C_0 , where C_n is a certificate trusted by the verifier. Then certificate path verification, i.e., proving that all the certificates in the path are valid, is required for verifying C_0 . There are two ways to perform the verification process: one is *top-down* verification, and the other is *bottom-up* verification. In bottom-up verification, the verifier starts to consider the validity of C_0 . He first checks if $\overline{I}(C_0) \downarrow \overline{S}(C_0)$ holds, checks if $\overline{D}_s(C) \leq \text{Today} \leq \overline{D}_e(C)$ holds, and checks

if C_0 does not belong to $\eta_{\overline{I}(C_0)}$. If any of the three checks fails, the verification process terminates with the result that C_0 cannot be accepted as valid from this path. If all of these checks are successful, the verifier then checks validity of the signature on C_0 . He then needs to consider the validity of C_1 . In the same way, he may terminate with the result that C_0 cannot be accepted as valid from this path because C_1 is not valid, or proceed to consider the validity of C_2 . Continuing the process, at the last, the verifier may only need to consider the validity of C_n . Because C_n is a certificate trusted by the verifier, $\text{valid}(C_n)$ holds. Thus, the verifier has proved that all certificates on the path are valid, and may therefore accept the certificate C_0 as valid.

With top-down verification, the verifier starts to consider the validity of C_{n-1} by accepting C_n as a valid certificate, because he trusts C_n .

6 Conclusion

We have presented an approach to the formal specification of the structure, functions and the security policy of a CMS. A state-based model for CMSs has been proposed. With this model, all CMS functions, such as certificate issuing, certificate revocation and certificate rekeying, have formally been specified as transitions that change the states of a CMS. For the major CMS client function – certificate verification, we have in particular discussed the verification principle, certificate path development, and given the verification rules which can be used for certificate path verification.

The correctness and effectiveness of security mechanisms for a system (particularly a computer network) usually depend on the understanding of the developer and the evaluator to the security requirements of the system and their analysis. Many mechanisms, presented in somewhat descriptive form, are not suitable for formal analysis and practical implementation. Our model provides an approach to the formal specification of a CMS, the security policy and CMS functions are all formally specified. Based on this model, the security mechanisms, which are used to enforce the policy, can also be presented in a formal form. Such a formal presentation can help the developer and the evaluator as well as the user to understand the security-relevant behavior of a system, and guide them in the design and analysis of the security architecture, and in the implementation of the system. The formal approach proposed in this paper and the state-based model itself may be used as the basis for other activities in the secure computer network area, such as research into new mechanisms, services and secure network applications areas, and integration of individual mechanisms into large secure and reliable network architectures.

As a future work, we have planned to mechanise our theory in a general theorem prover, Isabelle [9]. Once a reasoning system for CMSs has been developed, certificate verification and the proof of security properties of a CMS can be automatically performed.

Future work also includes investigating the certificate verification algorithms based on the state-based model. It may also be interesting to look at the different

distributions of trust points within a CMS in more detail. Joining CMS's, with so called cross-certificates, is another important issue. Comparisons of solutions and suggestions as to how distribution of trust points could be implemented in these extended CMS structures also needs to be considered.

Acknowledgements

We would like to thank Dr. Brendan Mahony and Dr. Jim McCarthy for helpful discussions on the theory and techniques aspects. Thanks are also due to anonymous referees for their valuable comments and suggestions.

References

1. K. A. Eastaughffe, M. A. Ozols, and A. Cant. Proof tactics for a theory of state machines in a graphical environment. In *Proceedings of the 14th International Conference on Automated Deduction (CADE-14)*, Lecture Notes in Artificial Intelligence, pages 366–379. Springer-Verlag, 1997. [80](#)
2. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transaction on Information Theory*, 31:469–472, 1985. [75](#)
3. W. Ford. Advances in public-key certificate standards. *ACM SIGSAC Security Audit & Control Review*, 13(3), 1995. [75](#)
4. W. Ford and M. Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption*. Prentice-Hall, 1997. [75](#)
5. R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. IETF X.509 PKI (PKIX) Working Group (Draft), January 1999. [76](#), [77](#)
6. N. Kapidzic. Extended certificate management system: Design and protocols. Technical report, DVS, 1997. [76](#)
7. N. Kapidzic. Creating security applications based on the global certificate management system. *Computers & Security*, 17:507–515, 1998. [76](#)
8. S. Kent. Privacy Enhancement for Internet Electronic Mail, Part II: Certificate-Based Key Management, Request for Comments 1422. Network Working Group, 1993. [77](#)
9. L. C. Paulson. *ML for Working Programmer*. Cambridge University Press, 1991. [91](#)
10. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998. [76](#)
11. Denis Trcek. Organization of certification authorities in a global network. *Computer Security Journal*, 10(1):72–81, 1994. [76](#)
12. T. Y. C. Woo and S. S. Lam. Authorization in distributed systems: A new approach. *Journal of Computer Security*, pages 107–136, 2(1993). [76](#)

Confidence Valuation in a Public-Key Infrastructure Based on Uncertain Evidence

Reto Kohlas and Ueli Maurer

Department of Computer Science
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland
`{kohlas,maurer}@inf.ethz.ch`

Abstract. Public-key authentication based on public-key certificates is a special case of the general problem of verifying a hypothesis (that a public key is authentic), given certain pieces of evidence. Beginning with PGP, several authors have pointed out that trust is often an uncertain piece of evidence and have proposed ad hoc methods, sometimes referred to as trust management, for dealing with this kind of uncertainty. These approaches can lead to counter-intuitive conclusions as is demonstrated with examples in the PGP trust management. For instance, an introducer marginally trusted by a user can make him accept an arbitrary key for any other user.

In this paper we take a general approach to public-key authentication based on uncertain evidence, where not only trust, but also other pieces of evidence (e.g. entity authentication) can be uncertain. First, we formalize the assignment and the valuation of confidence values in the general context of reasoning based on uncertain evidence. Second, we propose a set of principles for sound confidence valuation. Third, we analyze PGP and some other previous methods for dealing with uncertainty in the light of our principles.

Keywords: Public-key certification, public-key infrastructure (PKI), web of trust, Pretty Good Privacy (PGP), evidence theory, reasoning with uncertainty.

1 Introduction

1.1 Motivation

Public-key cryptography is a basic technology for information security and electronic commerce. A prerequisite for the application of public-key cryptography is that the keys are authenticated. The validation of public keys is hence of paramount importance.

This is achieved by public-key certificates. A certificate is a digitally signed statement by which a certification authority (e.g., called trusted third party or introducer) asserts that a key is bound to an entity. The term “binding” is ambiguous [8] but for the purpose of this paper this is not relevant. The term public-key infrastructure (PKI) is used to refer to the complete legal, technical and organizational framework for drawing conclusions from a given set of certificates, trust relations and other pieces of evidence.

A collection of certificates where for instance Bob certifies Carol's key and Carol certifies Dave's key (and so on) is called a *certification path*. Since a chain is at most as strong as the weakest link, PGP [20,22] introduced the use of parallel certification paths in order to improve the reliability of a decision about the authenticity of a public key. A certification structure with multiple certification paths is sometimes referred to as a *web of trust*.

One of the uncertainty factors in public-key authentication is the possible intentional or unintentional misbehavior of entities, either by not carefully authenticating a person before issuing a certificate, or by creating false certificates in the first place. Both trust and authentication are hence uncertain to some degree and should preferably be modeled as such. A user (say Alice) should be able to express that she considers one entity more trustworthy than another, or that she considers authentication by passports more secure than authentication by voice recognition over a telephone line.

A value assigned to a piece of evidence as a degree of confidence is called a *confidence value*. A confidence value can be a discrete numerical or non-numerical value, or it can be a real value which may or may not be interpreted as a probability. The problem of assigning and evaluating confidence values numerically or on a discrete scale (as in PGP) is non-trivial, in particular when certification paths intersect, as will be illustrated.

In PGP 2.6.2 for example, a user Alice assigns a value from the set `{unknown, no trust, marginally trusted, fully trusted}` to every key K she retrieved from the PKI. This trust value for K implicitly stands for her trust in the entity that presumably controls K . All keys generated by Alice herself are by default authentic (or `valid`). To determine the validity of a key K , only the signatures under K generated with `valid` keys are considered. A key is accepted to be `valid` if it is signed at least by one `fully trusted` key, or by two `marginally trusted` keys.¹

Figure 1 shows two PGP-like webs of trust in a graphical notation introduced by Stallings [20]. A circle stands for an entity-key pair, and an arrow from A to B means that A's public key has been signed by B's public key. In the left scenario of Figure 1 for instance, X_1 has issued a certificate asserting that a certain public key is controlled by X_3 . Different patterns indicate the different trust values that an entity (in our examples Alice) assigned to a key. In the left scenario for instance, Alice `fully trusts` X_1 and X_2 , and she `marginally trusts` X_3 and X_4 .

In PGP 2.6.2, in the left scenario Bob's key is accepted to be `valid` while it is not in the right scenario. Although in the right scenario X_3 and X_4 are `fully trusted`, Bob's key is `not valid`, since already the keys X_3 and X_4 are `not valid`. One can argue that the two scenarios are isomorphic in a certain sense and that therefore it is counter-intuitive that the validity of Bob's key is different. In both scenarios, the same coalitions of misbehaving entities can cause

¹ This is a simplified description of PGP's trust management. For example, Alice can choose an upper bound for the length of certification paths to be considered. In the newer PGP releases, another trust management is implemented.

Alice to accept a false key for Bob: any one of the sets $\{X_1, X_2\}$, $\{X_1, X_3\}$, $\{X_2, X_4\}$ and $\{X_3, X_4\}$ (or of course a superset thereof). The two cases are isomorphic in the sense that in each case, one of the sets consists of two **fully trusted** entities, two sets consist of one **fully trusted** and one **marginally trusted** entity, and one set consist of two **marginally trusted** entities. It is one of the goals of this paper to formalize such principles like the described isomorphism.

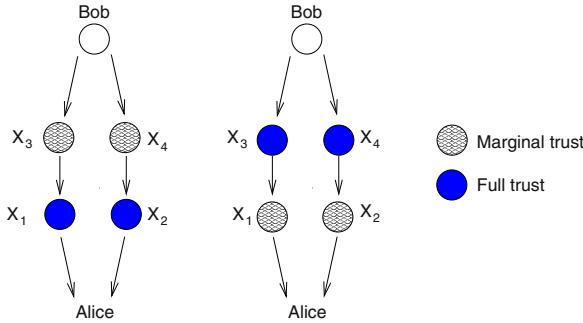


Fig. 1. Examples of public-key rings in PGP.

1.2 Contributions of This Paper

In this paper we take a general approach to public-key authentication based on uncertain evidence, where not only trust, but also other pieces of evidence (e.g. entity authentication) can be uncertain. Two papers with similar goals are [11] and [17].

Generic techniques for reasoning with uncertainty have been proposed in artificial intelligence (e.g., automated medical diagnosis) and decision theory (see, for instance, [13, 19, 9, 12, 6, 10]). Many techniques and approaches for reasoning with uncertain information are based on probability theory [13, 12, 6]. There are at least two conceptually different approaches to using probabilities in this context. In the first approach, probabilities are related to measured or estimated relative frequencies of an event under consideration (e.g. the relative frequency of a disease, given the particular evidence and symptoms observed on a patient), and such systems are often called expert systems.

In the second approach, probabilities are used as parameters of subjective belief, without necessarily having a direct interpretation as relative frequencies.² For instance, if one assigns a trust value of, say, 80% to a person, this generally

² In fact, the initial motivation (e.g. by Bernoulli [2]) for introducing probabilities was to argue about degrees of belief, in particular in the context of weighting statements made by different witnesses, where one cannot define an experiment in which relative frequencies make sense.

does not mean that one expects the person to misbehave 20% of the time. Nevertheless, interpreting this parameter as a probability of 0.8 makes sense. This is a point that is often misunderstood [3, 16, 14]. Of course, the parameters are generally based also on some form of past experience, but such experience almost never arises in the form of relative frequencies. Moreover, while confidence parameters in expert systems can often be verified in the real world, this is not the case for this second type of probability-based parameters.

Hence we disagree at a conceptual level with the approach taken by Beth, Borchering and Klein [3] and principle 2 of Reiter and Stubblebine [16, 14]: it is not necessary to define the parameters (i.e., the confidence values) of an authentication method as “negative and positive experiences” or frequencies.

We formalize the assignment and the valuation of confidence values in the general context of reasoning based on uncertain evidence. From an abstract point of view, in any uncertainty method, an entity assigns confidence values to the pieces of evidence she collected; these confidence values stand for her degree of belief that the corresponding piece of evidence is true. We then propose a set of principles for sound confidence valuation. These principles describe how a confidence valuation should reduce the confidence values assigned to the pieces of evidence to a single confidence value for the hypothesis under consideration. Two key concepts in the characterization of the confidence valuation are the notions of assumptions and arguments. These concepts are borrowed from the so-called argumentation systems [7, 5]. Finally, we analyze PGP and some other previous methods for dealing with uncertainty in the light of our principles.

While the main contribution of the paper is on modeling uncertainty, we also make some observations regarding the modeling of evidence in the context of PKI’s which are valid regardless of whether one considers evidence to be uncertain.

1.3 Previous Work and Outline

The design of methods for valuating the authenticity of public keys has received substantial attention in the cryptographic community (see [21, 22, 3, 15, 16, 11, 17, 14]); most of these methods are ad hoc and will not be discussed here in detail. Reiter and Stubblebine stated guidelines for a “metric” of authentication ([16]) which will be discussed in the concluding section.

In Section 2 we discuss various ways of modeling and dealing with uncertainty. Key concepts of propositional logic and argumentation systems are revisited. We formalize the concept of an uncertainty method by introducing the notions of confidence value, confidence assignment and confidence valuation. In Section 3 we state desirable principles for confidence valuation. In Section 4 we analyze existing confidence valuation methods in the light of our principles. We show some problems arising in PGP’s method for combining trust values. Finally, in Section 5, we compare our work with the principles of Reiter-Stubblebine [16] and mention some directions for future research.

2 Reasoning with Uncertainty

A *hypothesis* h is a statement for which one generally does not know whether it is true or not. In order to evaluate the truth of h , one can exploit dependencies of h on other facts whose truth one can observe or estimate. Such other facts or observations are often called *pieces of evidence* or simply *evidence*.

2.1 Propositional Logic and Logical Consequence

Logic is about relating the truth of different units of concern to each other, and hence logic allows to describe the truth of a hypothesis in dependency of a user's (Alice's) evidence. In this paper, we will use propositional logic, but concepts such as logical consequence, assumptions and arguments could also be defined in the context of more powerful languages, for instance first-order logic (for an excellent introduction to different logics see [1]).

The basic units of concern in propositional logic are called *propositions* or, alternatively, *statements*. We denote the set of statements by \mathcal{S} , and statements by s, s_1, \dots . The statement standing for the hypothesis is sometimes denoted by h . A *formula* of propositional logic is composed of statements and *logical connectives*. In the sequel, let g, f, f_1, f_2, \dots stand for formulas. In a standard definition of propositional logic, there are three connectives: \neg , \wedge , \vee . “Implies” ($f \rightarrow g$) is a shorthand for $\neg f \vee g$.

A formula is either true or false, depending on the truth values that have been assigned to the statements. A *truth assignment* \mathcal{B} is a function from the set of statements \mathcal{S} to the set of truth values, $\{\text{true}, \text{false}\}$, often represented by 1 and 0: $\mathcal{B} : \mathcal{S} \rightarrow \{0, 1\}$. The semantic definition of propositional logic states how the different logical connectives combine the truth of the corresponding subformulas, i.e., what the truth value of a formula f , denoted by $\hat{\mathcal{B}}(f)$, is. The logical connective \neg stands for “not”: the formula $\neg f$ is true only if f is false. The formula $f \wedge g$ is true if f is true *and* g is true, and $f \vee g$ is true if f is true *or* g is true. A truth assignment \mathcal{B} such that the formula f is true (i.e., $\hat{\mathcal{B}}(f) = 1$) is called a *model* for f . A formula that has at least one model is called *satisfiable*, and otherwise *unsatisfiable*. A formula g is *logical consequence* of f (or f follows from g), if every model of f is also model of g . This is denoted by $f \models g$. If f and g have the same set of models, i.e. $f \models g$ and $g \models f$, then f and g are called *semantically equivalent*.

One can represent each piece of evidence and the hypothesis by a statement in \mathcal{S} , and one's belief is a formula Σ over \mathcal{S} . The hypothesis h is accepted if, whenever Σ is true, also h is true. This corresponds to $\Sigma \models h$.

2.2 Assumptions and Arguments

Pieces of evidence can also be uncertain, and not only the hypothesis. If a hypothesis h does not follow from an initial belief Σ there are sometimes assumptions one can make about the truth values of some uncertain pieces of evidence in order to derive h . Informally, such a combination of assumptions is called an

argument for h . Often there are different arguments for h ; the set of arguments is what one could call a qualitative characterization for the uncertainty of h .

The notions of assumptions and arguments as used in this paper have been introduced in the context of assumption-based truth maintenance systems (ATMS) [4], and formalized in the context of argumentation systems [5]. We will define assumptions and arguments similarly as it has been done in the case of argumentation systems. However, since we do not need the entire power of argumentation systems, we can use a simpler notation.

An *assumption* is a piece of evidence; we denote the set of pieces of evidence by \mathcal{E} , where $\mathcal{E} \subseteq \mathcal{S}$. The pieces of evidence, i.e. the assumptions are denoted by a, a_1, \dots

A *conjunction* is of the form $l_1 \wedge \dots \wedge l_m$, where the l_i are literals; a literal is a propositional statement s or its negation $\neg s$. A conjunction is non-contradicting, if no statement s occurs positively and negatively in the conjunction. In the sequel, let \mathcal{L}_A denote the set of literals over the set of assumptions \mathcal{E} and \mathcal{C}_A denote the set of non-contradicting conjunctions over \mathcal{E} . We write a conjunction $l_1 \wedge \dots \wedge l_m$ also as a set $\{l_1, \dots, l_m\} \subseteq \mathcal{L}_A$.

An *argument* \mathcal{A} for a hypothesis h is a non-contradicting conjunction over the set of assumptions \mathcal{E} (i.e., $\mathcal{A} \in \mathcal{C}_A$) such that h can be derived from $\mathcal{A} \wedge \Sigma$: $\mathcal{A} \wedge \Sigma \models h$.³ In fact, we could have also defined that an argument is any formula over \mathcal{E} . However, as shown below, the arguments for h with respect to Σ can always be represented by a set of so-called minimal non-contradicting conjunctions. Such a set of conjunctions for h is called an *argument structure* and is denoted by \mathcal{A}^* , where $\mathcal{A}^* \subseteq \mathcal{C}_A$.

Let \mathcal{A} be any propositional formula over \mathcal{E} , such that $\mathcal{A} \wedge \Sigma \models h$. \mathcal{A} is semantically equivalent to a formula $\mathcal{A}_1 \vee \dots \vee \mathcal{A}_n$, where the \mathcal{A}_i are conjunctions in \mathcal{C}_A . Every \mathcal{A}_i is also an argument for h , since the set of models for \mathcal{A}_i is contained in the set of models of $\mathcal{A}_1 \vee \dots \vee \mathcal{A}_n$. Furthermore, if the conjunction $\mathcal{A}_i = a_{i1} \wedge \dots \wedge a_{im}$ is argument for h , then so is the conjunction that is obtained by adding one assumption to \mathcal{A}_i . A formula $\mathcal{A} \in \mathcal{C}_A$ such that \mathcal{A} is satisfiable and $\mathcal{A} \wedge \Sigma$ is unsatisfiable is called an argument for the *contradiction*.

2.3 Argument Structures for Horn Formulas

In this paper, we investigate the special case where Σ is a so called *Horn formula*. A *Horn formula* is a conjunction $f_1 \wedge \dots \wedge f_n$ of *Horn clauses* f_i . A *Horn clause* f_i is a formula $s_1 \wedge \dots \wedge s_n \rightarrow s_{n+1}$, where the statements s_1, \dots, s_n are called the *preconditions* and s_{n+1} the *postcondition*. A Horn clause $s_1 \wedge \dots \wedge s_n \rightarrow s_{n+1}$ means that the statement s_{n+1} must necessarily be true if $s_1 \dots s_n$ are true.

In the following we prove some properties of the argument structure \mathcal{A}^* in the case that Σ is a Horn formula. In a first reading, the proofs can be skipped.

³ In the literature, such arguments are called *supporting* [5]. An argument \mathcal{A}_2 such that h is still possible, i.e. such that the counter-hypothesis $\neg h$ cannot be derived ($\mathcal{A}_2 \wedge \Sigma \not\models h$), is called a *plausible* argument.

Lemma 2. Let \mathcal{A} be a formula in \mathcal{C}_A . A statement $h \in \mathcal{S}$ can be derived from $\mathcal{A} \wedge \Sigma$, i.e., $\mathcal{A} \wedge \Sigma \models h$, if and only if $h \in \mathcal{A}$ or if there is a Horn clause $s_1 \wedge \dots \wedge s_m \rightarrow h$ in Σ such that $\mathcal{A} \wedge \Sigma \models s_i$, for $i = 1 \dots m$.

Proof. We first show that $f \models g$ if and only if $f \wedge \neg g$ is unsatisfiable. Consider any model \mathcal{B} for f . If $f \models g$, then \mathcal{B} is also a model for g . Hence $\hat{\mathcal{B}}(\neg g) = 0$ whenever $\hat{\mathcal{B}}(f) = 1$ and therefore $f \wedge \neg g$ is unsatisfiable. Conversely, that the formula $f \wedge \neg g$ is unsatisfiable means that if $\hat{\mathcal{B}}(f) = 1$ then $\hat{\mathcal{B}}(\neg g) = 0$. Hence whenever $\hat{\mathcal{B}}(f) = 1$ we have $\hat{\mathcal{B}}(g) = 1$. This corresponds to $f \models g$.

The problem of verifying $\mathcal{A} \wedge \Sigma \models h$ is therefore equivalent to deciding the unsatisfiability of $\mathcal{A} \wedge \Sigma \wedge \neg h$. The *marking algorithm* allows to determine the unsatisfiability of a formula $\mathcal{A} \wedge \Sigma \wedge \neg h$ [1]. Note that $\neg h$ is semantically equivalent to $h \rightarrow 0$, where 0 stands for an unsatisfiable formula.

Marking algorithm.

1. Rewrite $\mathcal{A} \wedge \Sigma \wedge \neg h$: Write all negative literals s_i in \mathcal{A} as $s_i \rightarrow 0$ and $\neg h$ as $h \rightarrow 0$.
2. Mark all statements that occur positively in \mathcal{A} .
3. While there is a Horn clause $s_1 \wedge \dots \wedge s_m \rightarrow B$ in Σ where the s_i for $i = 1 \dots m$ are marked and B is not yet marked. If $B = 0$, return **Unsatisfiable** and stop.
Else mark $B = s_{m+1}$.
4. Return **Satisfiable** and stop.

The time complexity of the algorithm is linear in the number of statements in \mathcal{S} : the while loop is executed at most $|\mathcal{S}|$ times. We show that the algorithm is correct: it returns **Unsatisfiable** if and only if $\mathcal{A} \wedge \Sigma \wedge \neg h$ is unsatisfiable. Thus h is logical consequence of $\mathcal{A} \wedge \Sigma$ if and only if $s \in \mathcal{A}$ or if there is a Horn clause $s_1 \wedge \dots \wedge s_m \rightarrow h$ such that $\mathcal{A} \wedge \Sigma \models s_i$, for $i = 1 \dots m$.

If the algorithm stops at step 3 and returns **Unsatisfiable** then the formula is unsatisfiable. For a model \mathcal{B} of $\mathcal{A} \wedge \Sigma \wedge \neg h$ (if there is at all any model) one must have $\mathcal{B}(s) = 1$ for all statements s that have been marked during the algorithm: All statements that occur positively in \mathcal{A} must be true and by definition of \rightarrow also the postcondition if all of its preconditions are true. The algorithm only returns **Unsatisfiable** if there is a Horn clause $s_1 \wedge \dots \wedge s_m \rightarrow 0$. In this case, because the s_i are marked, for all possible models \mathcal{B} of $\mathcal{A} \wedge \Sigma \wedge \neg h$, $\hat{\mathcal{B}}(s_1 \wedge \dots \wedge s_n \rightarrow 0) = 0$, and therefore $\mathcal{A} \wedge \Sigma \wedge \neg h$ indeed is unsatisfiable.

On the other hand, if the algorithm stops at step 4 and returns **Satisfiable**, a model for $\mathcal{A} \wedge \Sigma \wedge \neg h$ exists, and hence $\mathcal{A} \wedge \Sigma \wedge \neg h$ is satisfiable. A model \mathcal{B} is obtained by assigning the truth value 1 to all statements that have been marked, and 0 to the others. We have to show that $\hat{\mathcal{B}}(\mathcal{A} \wedge \Sigma \wedge (h \rightarrow 0)) = 1$. Clearly, $\hat{\mathcal{B}}(\mathcal{A}) = 1$: all positive statements have been marked (at step 2); the negative statements have not been marked, because otherwise the algorithm would have stopped in 3. The same reasoning can be applied to h : $\mathcal{B}(h) = 0$ and therefore $\hat{\mathcal{B}}(\neg h) = 1$. We get $\hat{\mathcal{B}}(\Sigma) = 1$. Let f denote any Horn clause in Σ , $f = s_1 \wedge \dots \wedge s_n \rightarrow s_{n+1}$. If s_{n+1} has not been marked, there is at least one s_j , j between 1 and n that has not been marked. Hence $\hat{\mathcal{B}}(f) = 1$ (and therefore $\hat{\mathcal{B}}(\Sigma) = 1$) since $\hat{\mathcal{B}}(s_j) = 0$ and $\hat{\mathcal{B}}(s_1 \wedge \dots \wedge s_n) = 0$. \circ

The proof of Lemma 2 shows that if $\mathcal{A} \wedge \Sigma \models h$, then this holds also for $\text{Pos}(\mathcal{A}) \wedge \Sigma \models h$, where $\text{Pos}(\mathcal{A})$ denotes the set of all positive literals occurring in \mathcal{A} . This implies that minimal arguments for h solely consist of positive literals.

Lemma 3. *Let $s_1 \wedge \dots \wedge s_n \rightarrow s_{n+1}$ be a Horn clause in Σ . If $\mathcal{A}_i \wedge \Sigma \models s_i$, for $i = 1 \dots n$, where the \mathcal{A}_i consist only of positive literals, then $\mathcal{A}_1 \wedge \dots \wedge \mathcal{A}_n$ is an argument for s_{n+1} : $\mathcal{A}_1 \wedge \dots \wedge \mathcal{A}_n \wedge \Sigma \models s_{n+1}$.*

Proof. First observe that $\mathcal{A}_1 \wedge \dots \wedge \mathcal{A}_n$ is satisfiable since the \mathcal{A}_i solely consists of positive literals. $\mathcal{A}_i \wedge \Sigma \models s_i$ implies that $\mathcal{A}_1 \wedge \dots \wedge \mathcal{A}_n \wedge \Sigma \models s_i$, for $i = 1 \dots n$. This means that for all models \mathcal{B} of $\mathcal{A}_1 \wedge \dots \wedge \mathcal{A}_n \wedge \Sigma$ we have $\mathcal{B}(s_i) = 1$. In this case, $\hat{\mathcal{B}}(s_1 \wedge \dots \wedge s_n \rightarrow s_{n+1}) = 1$ only if $\mathcal{B}(s_{n+1}) = 1$. Hence for all models of $s_1 \wedge \dots \wedge s_n \rightarrow s_{n+1}$ we have $\mathcal{B}(s_{n+1}) = 1$ and therefore $\mathcal{A}_1 \wedge \dots \wedge \mathcal{A}_n \wedge \Sigma \models s_{n+1}$. \circ

Lemma 2 and 3 implicitly describe how an argument for h can be determined. Take a Horn clause where h occurs as postcondition. Determine an argument \mathcal{A}_i for each of the preconditions s_i , $i = 1 \dots n$. If s_i is an assumption, an argument for s_i is given by s_i itself; if not, recursively try to find an argument for s_i . If for all s_i there is an argument \mathcal{A}_i , $\mathcal{A} = \wedge_{i=1}^n \mathcal{A}_i$ is an argument for h .

Lemma 4. *If $h \in \mathcal{S} - \mathcal{E}$ does not occur as a precondition of a Horn clause in Σ , then there are no arguments for the hypothesis $\neg h$, i.e., there is no $\mathcal{A} \in \mathcal{C}_A$, such that $\mathcal{A} \wedge \Sigma \models \neg h$.*

Proof. We have to show that there is at least one model \mathcal{B} for $\mathcal{A} \wedge \Sigma$ such that $\mathcal{B}(h) = 1$. By definition of an argument \mathcal{A} for h , there is at least one model for $\mathcal{A} \wedge \Sigma$; if $\mathcal{B}(h) = 1$, we are done. Otherwise, consider the truth assignment \mathcal{B}' , such that $\mathcal{B}'(h) = 1$, and $\mathcal{B}'(s) = \mathcal{B}(s)$ for all statements s expect h . \mathcal{B}' is also a model for $\mathcal{A} \wedge \Sigma$: $\hat{\mathcal{B}}'(\mathcal{A}) = 1$ (since h is not in \mathcal{A} , $\hat{\mathcal{B}}'(\mathcal{A}) = \hat{\mathcal{B}}(\mathcal{A})$). We have also that $\hat{\mathcal{B}}'(\Sigma) = 1$, since s occurs only as postcondition of a Horn clause. \circ

Lemma 5. *If no assumption $a \in \mathcal{E}$ occurs as postcondition of a Horn clause in Σ , then there are no arguments for the contradiction, i.e., $\mathcal{A} \wedge \Sigma$ is always satisfiable.*

Proof. Let $\{s_1, \dots, s_m\}$ be the set of all the m different statements occurring as postcondition of a Horn clause in Σ . By definition of an argument, \mathcal{A} has at least one model \mathcal{B} . Consider the truth assignment \mathcal{B}' : $\mathcal{B}'(a_i) = \mathcal{B}(a_i)$ for all assumptions in \mathcal{A} , $\mathcal{B}'(s_i) = \mathcal{B}(s_i)$ for $i = 1 \dots m$. Hence we have $\hat{\mathcal{B}}'(\mathcal{A}) = 1$ and $\hat{\mathcal{B}}'(\Sigma) = 1$. Since we can construct a model for $\mathcal{A} \wedge \Sigma$ for every $\mathcal{A} \in \mathcal{C}_A$ there is no argument for the contradiction. \circ

2.4 Degrees of Uncertainty

Any uncertainty method uses a partially ordered set of values to represent degrees of uncertainty (or belief). For a given assumption a , one's belief can range

from certainty that $\neg a$ is true over complete uncertainty to complete certainty that a is true. We assign confidence values only to non-negated assumptions, partial belief in a negated assumption can be represented by assigning a confidence value to a new non-negative assumption a' and introducing a new Horn clause $a' \rightarrow \neg a$.

Confidence values stand for the degree of certainty that a piece of evidence or a hypothesis is true. A *confidence set* \mathcal{T} is a partially ordered set of confidence values, where the ordering is denoted by $<$. The ordering of the confidence values indicate the degree of certainty; a higher confidence value stands for more certainty. As usual, $t_1 \leq t_2$ stands for $t_1 < t_2$ or $t_1 = t_2$.

A confidence set contains a minimal and a maximal confidence value, denoted by \perp and \top , respectively. The symbol \perp stands for complete uncertainty ($\perp \leq t, \forall t \in \mathcal{T}$). The confidence value \top stands for complete certainty ($t \leq \top, \forall t \in \mathcal{T}$) and captures an entity's belief that a statement is true.

A *confidence assignment* represents an entity's initial belief with respect to each of the assumptions. Formally, a confidence valuation is a function c from the set of pieces of evidence \mathcal{E} to the set of confidence values \mathcal{T} :

$$c : \mathcal{E} \rightarrow \mathcal{T}$$

Let \mathcal{C} denote the set of all confidence assignments. We assume that the confidence values assigned to the pieces of evidence are independent. Such an assumption is not restricting; in the approach described in this paper, dependencies between pieces of evidence could be encoded logically, i.e., as part of the evidence Σ . Consider the situation where the truth of the assumption a_1 depends on the truth of a_2 and vice versa. This dependency can be captured by introducing a statement (say a) and by replacing Σ by another formula Σ' : $\Sigma' = \Sigma \wedge (a \wedge a_1 \rightarrow a_2) \wedge (a \wedge a_2 \rightarrow a_1)$. The degree of dependency between a_1 and a_2 is then captured by the confidence value $c(a)$ assigned to a .

A *confidence valuation* e is a function that takes as input a hypothesis (i.e, a statement in \mathcal{S}) and a confidence assignment and returns a confidence value for the hypothesis.

$$e : \mathcal{C} \times \mathcal{S} \rightarrow \mathcal{T}$$

3 Principles for Confidence Valuation

A confidence valuation reduces the *a priori* information (the confidence values assigned to the pieces of evidence) to a single confidence value for the hypothesis. The principles of the next section characterize the way a confidence valuation should combine the confidence values assigned to the pieces of evidence in order to obtain a confidence value for the hypothesis.

In the following, let \mathcal{A}^* stand for the argument structure for h with respect to Σ . Our principles make sense if the argument structure \mathcal{A}^* has the following two properties:

1. There is no argument for the counter-hypothesis $\neg h$.

2. Every argument \mathcal{A} in \mathcal{A}^* consists solely of positive literals.

We will explain why these properties for \mathcal{A}^* are required when introducing our principles. Recall from Section 2 that if the hypothesis is a statement not belonging to the set of assumptions ($h \in \mathcal{S} - \mathcal{E}$) and if Σ is a Horn formula, then the argument structure has the above two properties.

If all arguments for h consist of at least one assumption which is completely uncertain for Alice, then Alice will be completely uncertain with respect to the truth of h . Conversely, if Alice is completely certain about the truth of all assumptions of one argument for h , then Alice will also be certain about the truth of h .

Principle 1. (Meaning of \perp and \top .) If for all arguments $\mathcal{A}_i \in \mathcal{A}^*$ there is at least one assumption $a_{ij} \in \mathcal{A}_i$ such that $c(a_{ij}) = \perp$, then

$$e(c, h) = \perp.$$

If there is at least one argument $\mathcal{A}_i \in \mathcal{A}^*$ such that for all assumptions a_{ij} in \mathcal{A}_i , $c(a_{ij}) = \top$, then

$$e(c, h) = \top.$$

If one increases the confidence value for one piece of evidence then the confidence valuation should not return a lower confidence value for h :

Principle 2. (Monotonicity of e with respect to the confidence assignments.) Let c_1 and c_2 be two confidence assignments such that $c_1(a) \leq c_2(a)$ for all $a \in \mathcal{E}$. Then,

$$e(c_1, h) \leq e(c_2, h).$$

Note that this principle makes only sense if the arguments in \mathcal{A}^* do not contain negative literals. If Alice's confidence in a increases, then Alice has less confidence in $\neg a$. If $\neg a$ is in an argument for h , then the hypothesis h is less supported and therefore h is less certain; therefore the result of the confidence valuation should decrease.

Two argument structures \mathcal{A}_1^* and \mathcal{A}_2^* are called isomorphic if the assumptions in \mathcal{A}_1^* can be renamed such that \mathcal{A}_2^* is identical to \mathcal{A}_1^* (in the sense of equality between sets). The notion of isomorphism captures our intuition in which case two argument structures for two hypotheses can be regarded to be equally strong: this is the case if \mathcal{A}_1^* and \mathcal{A}_2^* are equal up to the names that have been chosen for the assumptions (or more generally for the statements).

Definition 1. Let

$$\mathcal{A}_1^* = \{\{s_{11}^1, \dots, s_{1h}^1\}, \dots, \{s_{m1}^1, \dots, s_{mi}^1\}\}$$

and

$$\mathcal{A}_2^* = \{\{s_{11}^2, \dots, s_{1j}^2\}, \dots, \{s_{m1}^2, \dots, s_{mk}^2\}\}$$

be two argument structures for two statements h_1 and h_2 in $\mathcal{S} - \mathcal{E}$, respectively. \mathcal{A}_1^* and \mathcal{A}_2^* are isomorphic with respect to the function $f: \mathcal{E} \rightarrow \mathcal{E}$ if f is a bijection and

$$\begin{aligned} & \{\{f(s_{11}^1), \dots, f(s_{1h}^1)\}, \dots, \{f(s_{m1}^1), \dots, f(s_{mi}^1)\}\} = \\ & \quad \{\{s_{11}^2, \dots, s_{1j}^2\}, \dots, \{s_{m1}^2, \dots, s_{mk}^2\}\}. \end{aligned}$$

Assume that there are two isomorphic argument structures \mathcal{A}_1^* and \mathcal{A}_2^* for two hypotheses h_1 and h_2 , and let f be the isomorphism. Clearly, if the argument structure for the two hypotheses are isomorphic and if Alice's confidence is equal for all assumptions that correspond to each other, then the result of the confidence valuation should in both cases be the same.

Principle 3. (Isomorphism of argument structures.) Let \mathcal{A}_1^* and \mathcal{A}_2^* be two isomorphic argument structures for two hypotheses h_1 and h_2 , respectively. Let f denote the corresponding bijection. If for all $a \in \mathcal{E}$,

$$c_1(a) = c_2(f(a))$$

then

$$e(c_1, h_1) = e(c_2, h_2).$$

Consider two arguments \mathcal{A}_1 and \mathcal{A}_2 for h where $\mathcal{A}_1 \subset \mathcal{A}_2$. (Recall that arguments are represented as sets). Intuitively, \mathcal{A}_1 is a stronger argument than \mathcal{A}_2 since in the case of \mathcal{A}_1 , less assumptions must be true such that one can derive h . Formally, $\mathcal{A}_1 \subset \mathcal{A}_2$ implies that if $\mathcal{A}_1 \wedge \Sigma \models h$ then also $\mathcal{A}_2 \wedge \Sigma \models h$.

The next principle states that if for all arguments \mathcal{A}_1 in \mathcal{A}_1^* we can find a stronger argument \mathcal{A}_2 in \mathcal{A}_2^* , then the output of the confidence valuation should be equal or higher in the latter case.

Principle 4. (Implication.) Let \mathcal{A}_1^* and \mathcal{A}_2^* be the two argument structures for two hypotheses h_1 and h_2 , respectively. If for all argument $\mathcal{A}_1 \in \mathcal{A}_1^*$ there is an argument $\mathcal{A}_2 \in \mathcal{A}_2^*$ such that $\mathcal{A}_1 \supseteq \mathcal{A}_2$ then, for any confidence assignment c ,

$$e(c, h_1) \leq e(c, h_2).$$

4 Valuating Public-Key Authenticity

4.1 Modeling Public-Key Certification

In our approach, modeling a certain problem consists of identifying the pieces of evidence (i.e., \mathcal{E}), the possible conclusions (i.e., \mathcal{S}), and describing how the truth values of the evidence and the conclusion is related (i.e., Σ).

Different authentication methods are based on different models. The model of Reiter and Stubblebine consists of a set of public-key certificates [16]. PGP's method takes trust values with respect to keys into account, and in Maurer's model one assigns confidence values for the trustworthiness of a person [11] (see also below).

Since authenticating public keys is uncertain also because entities in a web of trust can misbehave (by issuing “wrong” certificates) an authentication method must rely on trust values that one assigns to persons. The problem that arises if trust is not with respect to persons but with respect to keys is best illustrated by the simple example depicted in Figure 2. In the left scenario Bob's key is accepted to be **valid** while in the right scenario it is not. Even if Carol is only **marginally trusted**, she can make Alice accept Bob's key to be **valid** by issuing certificates with two different keys. This is against the intention of the designer of PGP's method that two introducers are needed if they are only **marginally trusted**.

In the left scenario the assertions made by means of the public-key certificates and signed with K_1 and K_2 are treated as if they were independent whereas clearly they are not. K_1 and K_2 are both controlled by Carol and hence both certificates for Bob's key have been issued by Carol. If trust values would be assigned to persons, both certification path (and hence both arguments) would depend on the trust value assigned to Carol.

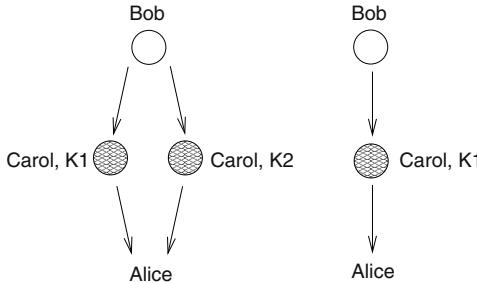


Fig. 2. A marginally trusted entity using multiple keys.

The same remark holds for the confidence valuations proposed by Reiter and Stubblebine [16]. A single person can produce any result for the confidence valuation, by generating an arbitrary number of keys and then by issuing public-key certificates.

4.2 Confidence Valuation in PGP 2.6.2

We first formalize PGP's authentication method. The evidence and the conclusions consist of three types of statements, which we denote by $Aut_{X,K}$, $Cert_{K_1,X,K_2}$ and $Trust_{X,K}$. $Aut_{X,K}$ stands for the fact that the signature public

key K is authentic for the entity X . $Cert_{K_1, X, K_2}$ means there exists a public-key certificate, signed with the signature key K_1 , claiming that K_2 is authentic for X . Finally, $Trust_{X, K}$ stands for the fact that X is trustworthy to provide authentic public keys of other entities by means of a public key K . A key K is authentic for a person X if there is a certificate issued by Y by means of a public key K_1 , and if Y is trustworthy to issue public-key certificates. Therefore Σ consists of Horn clauses of the following form:

$$Aut_{Y, K_1} \wedge Trust_{Y, K_1} \wedge Cert_{K_1, X, K} \rightarrow Aut_{X, K}$$

The confidence valuation has been informally described in Section 1 and is formalized in Appendix A.

PGP's confidence valuation follows principle 1. If on every path there is an entity whose trust value is **no trust** or if there is a key that is **not valid** then the target key is **not valid**. Conversely, if there is a path where all entities are **fully trusted** and where all keys are **valid** then the key is accepted to be **valid**.

Principle 2 is also satisfied by PGP's confidence valuation. If PGP evaluates a key to be **valid**, it will still be valid when Alice increases a confidence value for a statement in \mathcal{E} , that is if Alice modifies her public-key ring in the following way: Alice increases a trust value, signs a public key with her own key, or adds a public-key certificate.

Principle 3 is not met by PGP. Again, consider the two scenarios of Figure 1 and assume that the entity X_i “presumably” controls the key K_i . Here, the evidence consists of the following statements:

$$\begin{aligned} \mathcal{E} = \{ & Aut_{X_1, K_1}, Aut_{X_2, K_2}, Cert_{K_1, X_3, K_3}, Cert_{K_2, X_4, K_4}, Cert_{K_3, B, K_B}, Cert_{X_4, B, K_B} \\ & Trust_{X_1, K_1}, Trust_{X_2, K_2}, Trust_{X_3, K_3}, Trust_{X_4, K_4} \}. \end{aligned}$$

Σ is obtained by instantiating the above Horn formula with the statements in \mathcal{E} :

$$\begin{aligned} \Sigma = (& Aut_{X_1, K_1} \wedge Cert_{K_1, X_3, K_3} \wedge Trust_{X_1, K_1} \rightarrow Aut_{X_3, K_3}) \wedge \\ & (Aut_{X_2, K_2} \wedge Cert_{K_2, X_4, K_4} \wedge Trust_{X_2, K_2} \rightarrow Aut_{X_4, K_4}) \wedge \\ & (Aut_{X_3, K_3} \wedge Cert_{K_3, B, K_B} \wedge Trust_{X_3, K_3} \rightarrow Aut_{B, K_B}) \wedge \\ & (Aut_{X_4, K_4} \wedge Cert_{K_4, B, K_B} \wedge Trust_{X_4, K_4} \rightarrow Aut_{B, K_B}). \end{aligned}$$

The confidence assignment can be read out from the web of trust. For instance, in the left scenario, we have the following confidence assignment c_l (where **ft** stands for **fully trusted** and **mt** for **marginally trusted**):

Aut_{X_1, K_1}	Aut_{X_1, K_1}	$Trust_{X_1, K_1}$	$Trust_{X_2, K_2}$	$Trust_{X_3, K_3}$
val	val	ft	ft	mt
$Trust_{X_4, K_4}$	$Cert_{K_1, X_3, K_3}$	$Cert_{K_2, X_4, K_4}$	$Cert_{K_3, B, K_B}$	$Cert_{K_4, B, K_B}$
mt	val	val	val	val

An argument for $Aut_{X,K}$ simply corresponds to the set of statements that are on the paths from the source key (in our examples Alice) to the target key (which is allegedly controlled by X). Both scenarios of Figure 1 have the same argument structure for Aut_{B,K_B} , and the two argument structures are therefore isomorphic. The two arguments for Aut_{B,K_B} are

$$\begin{aligned}\mathcal{A}_1 &= \{Aut_{X_1,K_1}, Cert_{K_1,X_3,K_3}, Cert_{K_3,B,K_B}, Trust_{X_1,K_1}, Trust_{X_3,K_3}\}, \\ \mathcal{A}_2 &= \{Aut_{X_2,K_2}, Cert_{K_2,X_4,K_4}, Cert_{X_4,B,K_B}, Trust_{X_2,K_2}, Trust_{X_4,K_4}\}.\end{aligned}$$

There is a bijection $f : \mathcal{E} \rightarrow \mathcal{E}$:

$$\frac{\begin{array}{|c|c|c|c|c|} \hline Trust_{X_1,K_1} & Trust_{X_2,K_2} & Trust_{X_3,K_3} & Trust_{X_4,K_4} & f(x) \\ \hline \end{array}}{\begin{array}{|c|c|c|c|c|} \hline Trust_{X_2,K_2} & Trust_{X_1,K_1} & Trust_{X_4,K_4} & Trust_{X_3,K_3} & x \text{ else.} \\ \hline \end{array}}$$

such that $c_l(a) = c_r(f(a))$. Thus, according to Principle 3, one could postulate that the confidence valuation should return the same confidence value for both scenarios. However, this is not the case, as mentioned in Section 1.

Assume that for every argument \mathcal{A}_1 of a hypothesis Aut_{X_1,K_1} there is an argument \mathcal{A}_2 for Aut_{X_2,K_2} such that $\mathcal{A}_1 \supseteq \mathcal{A}_2$. This means that for every path \mathcal{A}_1 of K_1 there is a path \mathcal{A}_2 for K_2 such that \mathcal{A}_2 is a sub-path of \mathcal{A}_1 . In this case, the key K_1 is not accepted to be **valid** if the key K_2 is not accepted to be **valid**. Hence principle 4 is followed by PGP.

4.3 Maurer's Confidence Valuation

Maurer's model of a public-key infrastructure consists of two parts [11]. In the deterministic part, he identifies the pieces of evidence that a user allow to derive that a public key is authentic for a certain entity. The deterministic part corresponds to what we here call the model of the confidence valuation (see Subsection 4.1). In the probabilistic part, probabilities stand for degrees of uncertainty. His method is based on a well-defined random experiment. The probabilistic method is inspired from other uncertainty methods, where probabilities stand for degrees of uncertainty or subjective belief [18, 5].

Maurer uses a similar set of statements as PGP. In his model, however, trust is with respect to persons and not with respect to keys. Secondly, recommendations are part of the model: by means of a digitally signed statement, introducers can not only assert that a certain public key is authentic for a certain entity, but they can also recommend other entities to be trustworthy. For simplicity, we will describe a version of his model where we do not consider recommendations. The observations that we will make are nevertheless valid for the complete model.

The model consists of the following types of statements [11]. $Aut_{A,B}$ stands for A 's belief that she holds an authentic public key of B . $Trust_{A,B}$ means that A believes that B is trustworthy. $Cert_{X,Y}$ stands for the fact that X has issued a public-key certificate for Y .

The fact that only entities and not keys are parameters in Maurer's statements can raise confusion and has been criticized by Reiter and Stubblebine [16]. As they state, “entities don't sign certificates, keys do” (principle 1 in [16]). In

Maurer's model there is the implicit assumption that every entity controls only one key; therefore it is not explicitly mentioned which key is concerned. For instance, the statements $Cert_{X_1,Y}$ and $Cert_{X_2,Y}$ stand for the fact that X_1 and X_2 have issued a certificate for the same key of Y . This confusion could be avoided by explicitly introducing statements where the keys that are concerned are mentioned, as it is the case in our formalization of PGP. One would also obtain a more realistic model where an entity can hold more than one key.

$View_A$ is the set of pieces of evidence that A collected. From $View_A$, Alice tries to derive statements by recursively applying the following inference rule:

$$Aut_{A,X} \wedge Trust_{A,X} \wedge Cert_{X,B} \vdash Aut_{A,B}$$

The statement $Aut_{A,B}$ can be derived if $Aut_{A,X}$, $Trust_{A,X}$ and $Cert_{X,B}$ are in $View_A$ or, recursively, if they are derivable by applying the inference rule. $\overline{View_A}$ stands for the set of statements that are initially in $View_A$ or that can be derived from $View_A$.

Note that this derivation procedure is purely syntactic, and that Maurer therefore uses the symbol \vdash rather than \rightarrow to denote that a statement follows from a set of other statements. The notion of derivability in Maurer's model corresponds to our notion of logical consequence, in the following way. $View_A$ corresponds to \mathcal{E} and $\overline{View_A}$ to \mathcal{S} . Σ is obtained by instantiating the above inference rule with the statements in \mathcal{S} . As one can show, the statement s is derivable from $View_A$ (i.e., $s \in \overline{View_A}$) if and only if $\mathcal{E} \wedge \Sigma \models s$. A minimal set of statements \mathcal{V} such that the statement s can be derived from \mathcal{V} is called a path. Obviously, the notion of a path corresponds to our notion of a minimal argument, and the set of paths of $Aut_{A,B}$ corresponds to the argument structure.

In the sequel, let \mathcal{S}_A stand for the set of statements that are in Alice's view. In the probabilistic part, $View_A$ is interpreted as a random variable where the domain is the powerset of \mathcal{S}_A , i.e., $View_A$ takes as values subsets of \mathcal{S}_A . Alice expresses her uncertainty towards the pieces of evidence by specifying a probability distribution for $View_A$. In order to keep the number of confidence values that Alice must assign reasonably small, one can make the assumption that the pieces of evidence are independent. This means in particular that the introducers are assumed not to collude. In case of the independence assumption, Alice assigns a probability to each piece of evidence.

Since $View_A$ is a random variable, also $\overline{View_A}$ is a random variable. The confidence value for a hypothesis h is defined as the probability that h can be derived from $View_A$:

$$e(c, h) = P(h \in \overline{View_A}).$$

To compute $e(c, h)$, one can determine all minimal arguments for h . Let \mathcal{V}_i , $i = 1 \dots k$, stand for the k minimal arguments. $\mathcal{V}_i \subseteq View_A$ stands for the event that h can be derived from \mathcal{V}_i . The confidence value $e(c, h)$ is obtained by computing the probability that h can be derived from at least one argument, i.e., by computing the probability of the union of the events $\mathcal{V}_i \subseteq View_A$,

i.e.,

$$e(c, h) = P\left(\bigvee_{i=1}^k (\mathcal{V}_i \subseteq \text{View}_A)\right).$$

The probability $P(\mathcal{V}_i \subseteq \text{View}_A)$ is the product of the probabilities of the statements in \mathcal{V}_i , in case these probabilities are independent. Since the events $\mathcal{V}_i \subseteq \text{View}_A$ intersect, one cannot simply add up the probabilities $P(\mathcal{V}_i \subseteq \text{View}_A)$. A naive approach would be to establish a table where each row is indexed by a subset of \mathcal{S}_A . For each row $\mathcal{V} \subseteq \mathcal{S}_A$, one would store

$$P(\text{View}_A = \mathcal{V}) = \prod_{s \in \mathcal{V}} p(s) \prod_{s \notin \mathcal{V}} (1 - p(s)).$$

$e(c, h)$ is the sum of the probabilities of all \mathcal{V} from which s can be derived:

$$e(c, h) = \sum_{s \in \bar{\mathcal{V}}} P(\text{View}_A = \mathcal{V}).$$

More efficiently, the union of the events $P(\mathcal{V}_i \subseteq \text{View}_A)$ can be computed according to the inclusion-exclusion principle (see [11]).

Maurer's confidence valuation satisfies our principles.

Principle 1. By assumption for every argument \mathcal{V}_i there is a statement a such that $c(a) = 0$. Hence $P(\mathcal{V}_i \subseteq \text{View}_A) = 0$, for $i = 1 \dots k$. The probability of the union of events is 0 if the probability of all events is 0, and therefore $e(c, s) = P(\bigvee_{i=1}^k (\mathcal{V}_i \subseteq \text{View}_A)) = 0$.

Conversely, assume that there is an argument \mathcal{V}_i such that $c(a) = 1$ for all statements a in \mathcal{V}_i . Hence $P(\mathcal{V}_i \subseteq \text{View}_A) = 1$. Since the probability of the union of events is always bigger than the probability of one event we get $e(c, s) = 1$.

Principle 2. By increasing the probability of one statement a (i.e, $c'(a) > c(a)$) one gets $e(c, h) \geq e(c', h)$. Let \mathcal{V}_{-a} denote the set $\mathcal{V} - a$ and \mathcal{V}_{+a} denote the set $\mathcal{V} \cup a$. Observe that if h can be derived from \mathcal{V}_{-a} , then it can also be derived from \mathcal{V}_{+a} (but the inverse is not necessarily true). Therefore one can distinguish two cases. If h can be derived from \mathcal{V}_{-a} then $P(\text{View}_A = \mathcal{V}_{-a}) + P(\text{View}_A = \mathcal{V}_{+a})$ does not depend on the value $c(a)$. But there might exist rows such that h can not be derived from \mathcal{V}_{-a} but from \mathcal{V}_{+a} ; moreover the higher $c(a)$, the higher $P(\text{View}_A = \mathcal{V}_{+a})$.

Principle 3. Assume that the argument structures \mathcal{A}_1^* and \mathcal{A}_2^* for two hypotheses Aut_{A, B_1} and Aut_{A, B_2} are isomorphic with respect to f . Additionally, assume that there is twice the same probability distribution (i.e, $c_1(a) = c_2(f(a))$ for all $a \in \mathcal{E}$). Since the argument structures are isomorphic, one has to add up the same number of rows in order to compute the confidence value for Aut_{A, B_1} as for Aut_{A, B_2} . Moreover, since there is twice the same probability distribution, Maurer's confidence valuation will twice return the same result. Therefore it satisfies principle 3.

Principle 4. Assume that for every minimal argument \mathcal{V}_1 of Aut_{A, B_1} there is a minimal argument \mathcal{V}_2 of Aut_{A, B_2} such that $\mathcal{V}_1 \supseteq \mathcal{V}_2$. This implies that for

computing the confidence value of Aut_{A,B_2} one has to add up more rows than for Aut_{A,B_1} . Hence principle 4 is satisfied.

5 Conclusions

5.1 The Deficiency of Extensional Methods

One possible criterion to classify uncertainty methods is whether the uncertainty is dealt with *extensionally* or *intensionally* [13]. In extensional systems, the uncertainty of a formula is computed as a function of the uncertainty of its subformulas. In other words, the confidence value of a conclusion is a function of the confidence values of the preconditions of the rule [13]. In intensional systems, uncertainty is attached to “state of affairs” or “possible worlds”. There seems to be a trade-off between computational efficiency and semantic correctness: Extensional systems have the advantage of generally being computationally more efficient than intensional systems. On the other hand, extensional systems often suffer the deficiency to produce counter-intuitive conclusions [13].

PGP is a representative of an extensional system since the validity of a key is computed as a function of the trust values attached to the signature keys under the public key. From this perspective, it is not surprising that one can construct scenarios where PGP returns counter-intuitive results. Maurer’s approach is an example of an intensional system, because Alice specifies a probability distribution over her possible views, and the confidence value for the hypothesis is the probability that the hypothesis can be derived from the view.

5.2 The Principles of Reiter-Stubblebine

Reiter and Stubblebine also introduce principles for a method of authentication (the RS principles for short) [16], and it is appropriate to compare their work with ours. Their principles can be understood as general guidelines summarizing common sense, at the price of being vague, while our principles are formulated within a precise mathematical framework, hence more precise, at the price of being less comprehensive.

For instance, what it means for “the output of a metric to be intuitive” (in RS-principle 4) is made precise in this paper by the principles for the confidence valuation. As a second example, the RS-principle 5 (“A metric should be resilient to manipulation of its model by misbehaving entities, and its sensitivity to various forms of misbehavior should be made explicit.”) is also vague because it is not made precise what the model of a metric is and in what ways the model can be manipulated. We characterize the reliability of a web of trust by the argument structure for the given evidence and hypothesis. The RS-principles 1, 3, 5 and 6 concern the modeling of evidence rather than how to deal with uncertainty. RS-principle 7 (“A metric should be able to be computed efficiently.”) is again very general but of course everybody would agree to it.

5.3 A Direction for Future Research

Our principles are a first natural characterization of how a confidence valuation should combine initial confidence values. They allow to point out problems arising in PGP's trust management. It is not our claim that we provide a complete characterization of a confidence valuation. Figure 3 shows a pair of scenarios where PGP's confidence valuation apparently produces a counter-intuitive result, even if it does not violate any of our current principles. In the left scenario Bob's key is considered to be **invalid**, while in the right scenario it is considered to be **valid**. However, the left scenario seems to be more secure than the right one. In the left scenario X_3 and X_4 have to collude in order to palm off a wrong key for Bob on Alice, whereas in right scenario X_3 can achieve this alone. A research goal is to find a complete characterization of a confidence valuation; this is not only of interest in the context of applied cryptography, but more generally in artificial intelligence and evidence theory.

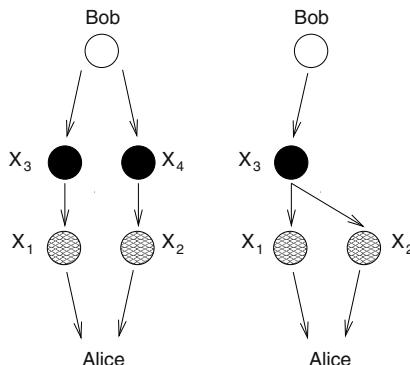


Fig. 3. Another inconsistence in PGP.

Acknowledgments

We wish to thank Christian Cachin, Rolf Haenni and Bartosz Przydatek for interesting discussions and helpful comments.

References

1. Nerode A. and Shore R. A.: *Logic for applications*. Springer-Verlag, 1993. 97, 99
2. J. Bernoulli. Ars conjectandi, 1713. Reprinted in 1968 by Culture et Civilisation, 115 Avenue Gabriel Lebon, Brussels. 95

3. T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open systems. In D. Gollmann, editor, *Computer Security - ESORICS'94*, volume 875 of *Lecture Notes in Computer Science*, pages 3–18. Springer Verlag, Berlin, 1994. [96](#)
4. J. de Kleer. An assumption-based TMS. *Artificial Intelligence, Elsevier Science Publisher B.V. (Amsterdam)*, 28:127–162, 1986. [98](#)
5. R. Haenni, J. Kohlas, and N. Lehmann. Probabilistic argumentation systems, 1999. [96](#), [98](#), [106](#)
6. M. Henrion, H. J. Suermondt, and D. E. Herckmann. Probabilistic and Bayesian representations of uncertainty in information systems: A pragmatic introduction. In Amihai Motro and Phillippe Smets, editors, *Uncertainty management in information systems*, chapter 9. Kluwer Academic Press, 1997. [95](#)
7. J. Kohlas and P.-A. Monney. A mathematical theory of hints. In *Lecture Notes in Economics and Mathematical Systems.*, volume 425. Springer, 1995. [96](#)
8. R. Kohlas and U. M. Maurer. Reasoning about public-key certification - on bindings between entities and public keys. In M. Franklin, editor, *Financial Cryptography 99*, LNCS, 1999. [93](#)
9. R. Kruse, E. Scheweke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge Based Systems*. Springer Verlag, 1991. [95](#)
10. E. H. Mamdani. On the classification of uncertainty techniques in relation to the application needs. In Amihai Motro and Phillippe Smets, editors, *Uncertainty management in information systems*, chapter 14. Kluwer Academic Press, 1997. [95](#)
11. U. M. Maurer. Modelling a public-key infrastructure. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Proceedings 1996 European Symposium on Research in Computer Security (ESORICS' 96)*, *Lecture Notes in Computer Science*, Springer, LNCS, pages 325–350, 1996. [95](#), [96](#), [104](#), [106](#), [108](#)
12. N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–86, 1986. [95](#)
13. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, Inc., 1988. [95](#), [109](#)
14. M. K. Reiter and S. G. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security*, 2(2), MAY 1997. [96](#)
15. M. K. Reiter and S. G. Stubblebine. Path independence for authentication in large-scale systems. *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 57–66, 1997. [96](#)
16. M. K. Reiter and S. G. Stubblebine. Toward acceptable metrics of authentication. *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 10–20, 1997. [96](#), [104](#), [106](#), [109](#)
17. A. Jøsang. An algebra for assessing trust in certification chains. In *Network and Distributed Systems Security (NDSS'99)*, 1999. [95](#), [96](#)
18. G. Shafer. Non-additive probabilities in the work of Bernoulli and Lambert. [106](#)
19. G. Shafer. *A mathematical Theory of Evidence*. Princeton University Press, 1996. [95](#)
20. W. Stallings. *Protect your privacy*. Prentice Hall, 1996. [94](#)
21. A. Tarah and C. Huitema. Associating metrics to certification paths. In *Computer Security ESORICS 92*, Lecture Notes in Computer Science, pages 175–189. Springer Verlag, Berlin, 1992. [96](#)
22. P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995. [94](#), [96](#)

A PGP's Confidence Valuation (Version 2.6.2)

Formally, PGP's confidence valuation can be described as follows:

$$e(c, Aut_{X,K}) = \begin{cases} \text{valid} & c(Aut_{X,K}) = \text{valid} \quad \text{or} \\ & \exists K_1, Y | c(Cert_{K_1,X,K}) = \text{valid} \wedge \\ & c(Trust_{Y,K_1}) = \text{ultimately trusted} \quad \text{or} \\ & \exists K_1, Y | c(Cert_{K_1,X,K}) = \text{valid} \wedge \\ & c(Trust_{Y,K_1}) = \text{fully trusted} \wedge \\ & e(c, Aut_{Y,K_1}) = \text{valid} \quad \text{or} \\ & \exists K_1, K_2, Y, Z | c(Cert_{K_1,X,K}) = \text{valid} \wedge \\ & c(Cert_{K_2,X,K}) = \text{valid} \wedge \\ & c(Trust_{Y,K_1}) = \text{marginally trusted} \wedge \\ & c(Trust_{Z,K_2}) = \text{marginally trusted} \wedge \\ & e(c, Aut_{Y,K_1}) = \text{valid} \wedge \\ & e(c, Aut_{Z,K_2}) = \text{valid} \\ \text{not valid} & \text{else.} \end{cases}$$

The Composite Discrete Logarithm and Secure Authentication

David Pointcheval

Dépt Informatique, ENS – CNRS, 45 rue d’Ulm, 75230 Paris Cedex 05, France.
`David.Pointcheval@ens.fr` – URL: <http://www.di.ens.fr/~pointche>.

Abstract. For the two last decades, electronic authentication has been an important topic. The first applications were digital signatures to mimic handwritten signatures for digital documents. Then, Chaum wanted to create an electronic version of money, with similar properties, namely bank certification and users’ anonymity. Therefore, he proposed the concept of blind signatures.

For all those problems, and furthermore for online authentication, zero-knowledge proofs of knowledge became a very powerful tool. Nevertheless, high computational load is often the drawback of a high security level. More recently, witness-indistinguishability has been found to be a better property that can conjugate security together with efficiency.

This paper studies the discrete logarithm problem with a composite modulus and namely its witness-indistinguishability. Then we offer new authentications more secure than factorization and furthermore very efficient from the prover point of view. Moreover, we significantly improve the reduction cost in the security proofs of Girault’s variants of the Schnorr schemes which validates practical sizes for security parameters. Finally, thanks to the witness-indistinguishability of the basic protocol, we can derive a blind signature scheme with security related to factorization.

1 Introduction

Provably secure schemes have ever been an important goal in cryptography. However, efficiency had hardly been an associated property. Even if authentication has been widely studied, very few schemes reach both efficiency and security. The reason is the large use of zero-knowledge protocols.

Identification. Concerning identification schemes, the first theoretical paper was the famous paper about zero-knowledge [19] which claimed that it was possible to prove the knowledge of a secret without revealing any information about it. Unfortunately, such a property, which guarantees security even against active attacks, often requires many iterations to actually reach a high security level and therefore results into inefficient protocols, either from the computational point of view [14,12,20,21,43,28,6,17] or from the communication load [45,49,50,31], and even both. Recently, a very efficient scheme has been proposed by Poupard and

Stern [38], with security relative to the discrete logarithm problem. However, the cost of the reduction is so high that the proof can not validate realistic parameters.

Few years ago, Feige and Shamir [13] defined weaker but sufficient properties for secure identification protocols, the “witness-hiding” and the “witness-indistinguishable” properties. They are indeed weaker than the zero-knowledge property in the sense that some information about the secret may be leaked, but not enough to efficiently find the secret. In other words, concerning the “witness hiding” property, if an attacker can find the secret after an active attack, she would have been able to find it without any interaction with the prover, within almost the same time. Whereas the “witness indistinguishable” property means that the view of the attacker is independent of the witness used as secret key: many secret keys are related to a public one and the proof only transfers the information that such a secret key is used, but not which one. In the following, we focus on this latter property which provides three-pass identification schemes secure against active attacks. Okamoto [26] presented some variants of the Schnorr [44] and Guillou-Quisquater [20] identification schemes, therefore related to the discrete logarithm in subgroups of prime order and to the RSA assumption [41] respectively.

The Random Oracle Model. For the last years, the so-called “random oracle model” [1] has boosted researches, providing an interesting tool for the designers since it helps to prove the security of very efficient schemes.

Indeed, this model, where some concrete cryptographic objects are idealized, namely the hash functions which are assumed to be really random ones, helped to provide security proofs for many encryption schemes [1,2,48,27,15,32,16,29,33] and digital/blind signature schemes [3,35,34,36,25,37], etc.

In spite of the recent paper [7] making people to be careful with the random oracle model, this latter is widely considered robust since it is more and more used. For example, the encryption scheme OAEP [2] which is proven secure in this model has been incorporated in SET, the Secure Electronic Transaction system [23] proposed by VISA and Master Card, and will become the new RSA encryption standard PKCS #1 v2.0 [42]. The security of many other schemes has been validated in this model.

1.1 Related Work

Identification and Signatures. Few years ago, Schnorr [43,44] presented a very efficient identification scheme, and the signature variant, based on the discrete logarithm problem in subgroups of prime order. We do not recall this famous scheme. However, the identification scheme is well-known to be zero-knowledge but only using a fixed-size challenge after many sequential iterations. Therefore, high security level against active attacks implies a high communication cost and either large memory for storing precomputations or high computation load, since no secure preprocessing has ever been proposed [10,11].

Nevertheless, many applications assume its security even with the basic three-pass protocol, using large challenges. Such a security would rely on the unproven assumption that this scheme is “witness-hiding”.

After the definitions of witness-hiding and witness-indistinguishable properties [13], Brickell and McCurley [6] proposed a variant of the Schnorr identification scheme dealing with the witness-hiding property. Then, Okamoto presented efficient three-pass identification schemes [26] provably secure even against active attacks, thanks to witness-indistinguishability. One of them uses the representation problem [4] and is therefore based on the discrete logarithm problem in subgroups of prime order. The second relies on the RSA assumption [41]. However, all of the above schemes remained less efficient than the original Schnorr’s scheme.

In 1991, Girault [17] presented a variant of the Schnorr identification scheme using a composite modulus instead of a prime one. It also allows an improved efficiency from the prover point of view. Few years ago, Poupard and Stern [38] provided a proof of the statistical zero-knowledge property of this scheme and the security relative to the composite discrete logarithm problem of both the identification scheme and the signature variant. However, the security proof, based on the zero-knowledge property of the identification scheme, also requires many iterations for a high level of security, and moreover uses an expansive reduction which can only validate large, and impractical, parameters. They recently improved their reduction [39,40], making security just relative to factorization. It is also the direction taken in the present work.

Concerning signatures, thanks to the Pointcheval–Stern’s [37] and Ohta–Okamoto’s [25] papers, one can efficiently transform any three-pass identification scheme into a signature scheme. Therefore, an efficient solution for identification furthermore solves the problem of efficient signatures.

Blind Signatures. In 1982, Chaum [8] wanted to create an electronic version of money, with similar properties, namely anonymity. He claimed that a way to do it was to use the notion of electronic coins together with blind signatures. A blind signature involves two participants, a user and the bank. The user wants to get a coin signed by the bank in such a way that the bank cannot recognize later either the coin nor the signature. He proposed a variation of the RSA signature [41] and later Brands [5] proposed a variation of the Schnorr’s one.

Unfortunately, none of those schemes admits any security proof. Excepted some theoretical propositions [9,30,22] which are totally impractical, we had to wait 1996 to see blind signature schemes [34] provably secure. They were based on the Okamoto [26] witness-indistinguishable protocols, and used the following functions, for which collisions are provably difficult to compute:

- *problem of representation = discrete logarithm:* $f_{p,g,h}(r, s) = g^r h^s \bmod p$.

A collision reveals the discrete logarithm of h in basis g . Indeed,

$$f_{p,g,h}(r, s) = f_{p,g,h}(r', s') \implies h = g^{(r'-r)/(s-s')} \bmod p.$$

- RSA problem/factorization: $f_{N,a,e}(r, s) = a^r s^e \bmod N$.

For some well-chosen parameters, a collision reveals the e -th root of a modulo N . Indeed,

$$f_{N,a,e}(r, s) = f_{N,a,e}(r', s') \implies a^{r'-r} = (s/s')^e \bmod N.$$

For a large enough prime e , Bezout's equality provides the e -th root of a modulo N . Otherwise, if e is a power of two and N a Blum integer, we can get the factorization of N (*cf.* [28,47]).

Later, another well-known witness-indistinguishable problem has been used [36], the *modular square root*: $f_N(x) = x^2 \bmod N$ for any $0 \leq x \leq N/2$, where

$$f_N(x) = f_N(y) \implies \gcd(N, x - y) \in \{\text{factors of } N\}.$$

In those papers, it was claimed that the proposed blind signature schemes were provably secure against parallel attacks. This means that the bank is guaranteed that after having given 10 dollars to a user, this latter cannot withdraw more than 10 dollars.

However, the main drawback of all those schemes is a high computation cost, even if they are practical, in comparison with the schemes claimed secure in the standard model [9,30,22]. It is, by now, an important challenge for blind signatures: a provably secure scheme which is also efficient, and particularly from the signer point of view since he may have thousands of signatures to perform at the same time.

1.2 Outline of the Paper

In this paper, we investigate, for the first time, the witness-indistinguishable protocols provided by the discrete logarithm problem with a composite modulus.

We first recall the Girault's scheme [17] together with the recent security results of Poupard and Stern [38]. Unfortunately, as for the Schnorr's scheme [43], this scheme has been proven zero-knowledge only using fixed-size challenges. Then many iterations are required to achieve a high security level. Here, we prove the security of this scheme, even against active attacks, after only one iteration of the protocol, using the witness-indistinguishable property [13]. That is an important improvement for the practical security w.r.t. the previous results [38]. Furthermore, we formally prove the security even if we use small keys, and thus for a very efficient scheme, whereas it was only heuristic. As previously said, the security of the signature is therefore a straightforward corollary [37] and can be considered as folklore.

Thereafter, we consider a blind signature scheme based on this problem, with a formal proof of security relative to factorization. Besides the provable security, the main property of this new scheme is efficiency, since it requires only one multiplication (not a modular one), from the computational point of view of the bank.

2 The Discrete Logarithm Problem

As shown by Feige and Shamir [13], the witness-indistinguishability (and even witness-hiding property) of an identification scheme is enough to provide security against active attacks. Pointcheval and Stern [37] proved that this property further provides blind signature schemes secure against one-more forgeries under parallel attacks.

The composite discrete logarithm problem provides such protocols, using the function $f_{N,g}(x) = g^x \bmod N$ for well-chosen N and g . Let us first define some useful notions for the following before stating an important theorem.

Definition 1 (α -strong prime). A prime integer p is said α -strong if $p = 2r + 1$ where r is a large integer whose prime factors are all greater than α .

Definition 2 (α -strong RSA modulus). An integer N is called an α -strong RSA modulus if $N = pq$ where p and q are both α -strong primes.

Definition 3 (asymmetric basis). Let $N = pq$ be an RSA modulus. A basis g in \mathbb{Z}_N^* is said asymmetric if the parities of $\text{Ord}(g)$ are different in \mathbb{Z}_p^* and \mathbb{Z}_q^* .

In other words, an asymmetric basis is a quadratic residue in only one of both subgroups \mathbb{Z}_p^* and \mathbb{Z}_q^* .

Theorem 4. Let $N = pq$ be any α -strong RSA modulus, for some $\alpha > 2$, and g any asymmetric basis in \mathbb{Z}_N^* , of order greater than α , then a collision of $f_{N,g}$, defined by $x \mapsto g^x \bmod N$, provides the factorization of N .

Proof. Let us denote by 2ℓ the order of g in \mathbb{Z}_N^* . One may remark that this order is necessarily even since it is even in at least, but also exactly, one of the subgroups, say \mathbb{Z}_p^* . Furthermore, ℓ is odd and greater than α , since it should be greater than $\alpha/2 > 1$ and any prime factor of $(p - 1)/2$ or $(q - 1)/2$ is odd and greater than α . Therefore,

$$g^{2\ell} = 1 \bmod p \text{ and } g^{2\ell} = 1 \bmod q, \text{ but } g^\ell = -1 \bmod p \text{ and } g^\ell = 1 \bmod q.$$

Let us assume that we have a collision $x < y$ for $f_{N,g}$, $f_{N,g}(x) = f_{N,g}(y)$. If we note $L = y - x$, then $2\ell|L$. By extracting the odd part b of L , $L = 2^a b$, we get a multiple of ℓ . Then

$$g^{2b} = 1 \bmod p \text{ and } g^{2b} = 1 \bmod q, \text{ but } g^b = -1 \bmod p \text{ and } g^b = 1 \bmod q.$$

Therefore, g^b is a non-trivial square root of 1 in \mathbb{Z}_N^* : $\gcd(g^b - 1, n) \in \{p, q\}$. \square

Then, we have a difficult problem for which two distinct solutions provide the factorization of the modulus N .

3 Application to Cryptographic Protocols

We first consider the identification scheme, together with the derived signature. Then, we focus on a new blind signature scheme.

3.1 Identification

Presentation. Let us first recall the Girault's scheme [17] (see Figure 1).

- We have two security parameters k and k' , where k represents the size of the challenge and k' is related to the information leak, and a bound S for the secret key. Then, we define $R = 2^{k+k'}S$. We use an RSA-modulus $N = pq$ and an element $g \in \mathbb{Z}_N^*$ of high order. The prover chooses a random secret key $s \in \{0, \dots, S - 1\}$ and publishes $v = g^{-s} \bmod N$.
- The prover initiates the protocol choosing a random $r \in \{0, \dots, R - 1\}$ and sending the “commitment” $x = g^r \bmod N$; The verifier randomly chooses a “challenge” $e \in \{0, \dots, 2^k - 1\}$, and sends it to the prover; Finally, the prover computes and sends $y = r + es$;
- The verifier can check whether $x = g^y v^e \bmod N$, or not.

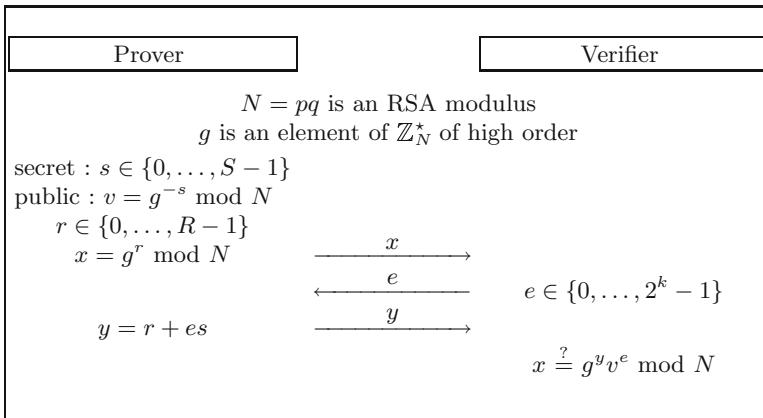


Fig. 1. Girault's Identification Scheme

One cannot say this is a proof of knowledge of the discrete logarithm of v in basis g modulo N , but one can state the following theorem in the particular case where N is a 2^k -strong RSA modulus (which includes the strong-RSA moduli classically used in practice):

Theorem 5. *Let N be a 2^k -strong RSA modulus. If there exists an attacker \mathcal{A} , with running time bounded by T , that is able to be accepted with a probability ε greater than $2 \cdot 2^{-k}$, for a non-negligible fraction of v , then discrete logarithms in basis g modulo N can be computed within an expected time bounded by $4T/\varepsilon \times S/\text{Ord}(g)$.*

Proof. Using the classical technique of extraction [14,12,37,25], one can obtain, from two valid proofs with the same commitment x , a pair (α, β) such that

$v^\alpha = g^\beta \bmod N$ with $0 < \alpha < 2^k$. Furthermore, this can be done in expected time bounded by $4T/\varepsilon$.

If one first runs this reduction with $v = g^\gamma \bmod N$, for a randomly chosen γ smaller than S , then one gets (α_1, β_1) such that

$$L = \alpha_1 \gamma - \beta_1 = 0 \bmod \text{Ord}(g),$$

which is nonzero with probability greater than $(S/\text{Ord}(g) - 1)^{-1}$.

Then, one runs this reduction with v whose discrete logarithm x is wanted and gets (α_2, β_2) . Let us initialize ℓ_0 to the L obtained above. Thereafter, we recursively compute $\ell_{i+1} = \ell_i / \gcd(\alpha_2, \ell_i)$ until the gcd equals 1. The limit is denoted by ℓ . Since $\alpha_2 < 2^k$, which is smaller than all the odd prime factors of $\lambda(N)$, and then of $\text{Ord}(g)$, 2ℓ is still a multiple of $\text{Ord}(g)$. We then compute $y = \beta \alpha_2^{-1} \bmod \ell$, and one gets $x = y + c\ell \bmod \text{Ord}(g)$, where $c \in \{0, 1\}$. One has just to check the right value for c .

One can remark that in the particular case where g is of maximal order $\lambda(N)$, a multiple of $\text{Ord}(g) = \lambda(N)$ leads to the factorization of N [24]. \square

Theorem 6. *This protocol is statistically zero-knowledge.*

Proof. The reader may refer to the Poupard-Stern's paper [38] or to the proof of witness-indistinguishability presented below. \square

However, the zero-knowledge property of an interactive proof of knowledge is a too strong property for identification purpose, and the main drawback is the sequential iterations of the basic scheme to achieve a high security level. Witness-indistinguishability [13] is therefore enough to ensure security against active attacks and provides a much more efficient scheme.

Theorem 7. *This protocol is statistically witness-indistinguishable.*

Proof. We have to prove that the distribution of the communication tapes is independent of the secret key used by the prover, even with a dishonest verifier. Let $s_1 < s_2$ be two distinct secret keys in $\{0, \dots, S - 1\}$ such that

$$g^{-s_1} = g^{-s_2} = v \bmod N.$$

We can show that the following distributions, where r is uniformly chosen in $\{0, \dots, R - 1\}$ and \mathcal{S} the strategy, possibly probabilistic, of the attacker to get the challenge e from x , are indistinguishable:

$$\begin{aligned} \delta_1 &= \{(x = g^r \bmod N, e, y) \mid y = r + s_1 e, e = \mathcal{S}(x)\} \\ \text{and } \delta_2 &= \{(x = g^r \bmod N, e, y) \mid y = r + s_2 e, e = \mathcal{S}(x)\}. \end{aligned}$$

Indeed, for any triple (α, β, γ) such that $\alpha = g^\gamma v^\beta \bmod N$, we can define

$$p_i(\alpha, \beta, \gamma) = \Pr_{(x,e,y) \in \delta_i} [(x, e, y) = (\alpha, \beta, \gamma)], \text{ for } i = 1, 2.$$

If we denote by $p_{\alpha,\beta}$ the probability for the strategy \mathcal{S} to output β on the input α , and if δ is the boolean function defined by $\delta(\text{true}) = 1$ and $\delta(\text{false}) = 0$, then we get

$$\begin{aligned} p_i(\alpha, \beta, \gamma) &= \Pr_r[\alpha = g^r \bmod N, \beta = \mathcal{S}(\alpha), \gamma = r + s_i \beta] \\ &= \Pr_r[\alpha = g^r \bmod N] \cdot p_{\alpha,\beta} \cdot \Pr_r[\gamma = r + s_i \beta | \gamma = r + s_i \beta \bmod \text{Ord}(g)] \\ &= \frac{1}{\text{Ord}(g)} \cdot p_{\alpha,\beta} \cdot \delta(0 \leq \gamma - s_i \beta < R) \cdot \frac{\text{Ord}(g)}{R}. \end{aligned}$$

An easy simplification leads to $p_{\alpha,\beta}/R \times \delta(s_i \beta \leq \gamma < R + s_i \beta)$. Therefore the distance between both distributions δ_1 and δ_2 is the sum over all the triples (α, β, γ) such that $\gamma = \log \alpha - s_1 \beta \bmod \text{Ord}(g)$:

$$\Delta = \sum_{\alpha,\beta} \frac{p_{\alpha,\beta}}{R} \cdot \frac{2(s_2 - s_1)\beta}{\text{Ord}(g)} \leq \frac{2S}{R \cdot \text{Ord}(g)} \times \sum_{\alpha,\beta} \beta \cdot p_{\alpha,\beta}.$$

By definition of the probability $p_{\alpha,\beta}$, it is clear that for any α , $\sum_{\beta} p_{\alpha,\beta} = 1$, and therefore the sum over all possible α is equal to $\text{Ord}(g)$. Since $\beta < 2^k$ and $R = 2^{k+k'}S$, we get

$$\Delta \leq \frac{2S \cdot 2^k}{R} = \frac{2}{2^{k'}}.$$

□

Thanks to this witness-indistinguishability, if we furthermore make g to be an *asymmetric basis* in \mathbb{Z}_N^* , we get an efficient and secure identification scheme in only three flows.

Theorem 8. *Let N be a 2^k -strong RSA modulus and g an asymmetric basis of high order in \mathbb{Z}_N^* . If $S \geq 2 \cdot \text{Ord}(g)$, this protocol is secure against active attacks relative to the factorization of N .*

Proof. In order to prove the security of the identification scheme against active attacks, we choose a random secret key $s < S$. We let the attacker verify some interactions. Then, we assume that she succeeds in her impersonation with probability ε . Using the first step of the proof of the Theorem 5, we get a multiple L of the order of g with probability greater than one half. Then, as in the proof of the Theorem 4, since g is an asymmetric basis, one gets the factorization of N . □

Remark 9. It is important to remark that we have only proven that an impersonation under an active attack is harder than the factorization, whereas the security of the iterated protocol has already been proven relative to the composite discrete logarithm, using the zero-knowledge property.

Nevertheless, when we have the factorization of N , the remaining security is the same as in the Schnorr's identification scheme [43,44]: heuristically, the discrete logarithm in subgroups of prime orders, which is still hard to solve.

Furthermore, the security result remains even if the challenge grows in order to get a high security level. Which is not the case for the zero-knowledge property.

Reduction Costs. Let us compare the reduction costs to obtain two answers from the impersonator. With the Poupart and Stern’s proof [38], this leads with some more computations to the discrete logarithm of v , with our proof, this immediately leads to the factorization of N .

Poupart and Stern’s Reduction. We assume that we use a k -bit challenge, and there exists an impersonator who succeeds with probability $\varepsilon \geq 2 \cdot 2^{-k}$. Then, one can easily show [25,37] that after less than $4/\varepsilon$ iterations, we get two valid answers with probability greater than $1/2$. As a consequence, a passive impersonation, with probability ε greater than $2 \cdot 2^{-k}$ within time T , can be used to find two distinct answers for a same commitment within an expected time bound $4T/\varepsilon$. On the other hand, an impersonation, with probability ε greater than $2 \cdot 2^{-k}$ after ℓ active attacks, within time T , can be used to find two distinct answers for a same commitment within a time bound $(4/\varepsilon + \ell \times 2^k) \cdot T$, because of the simulation which requires many resets. It is almost equal to $2^k \ell T$.

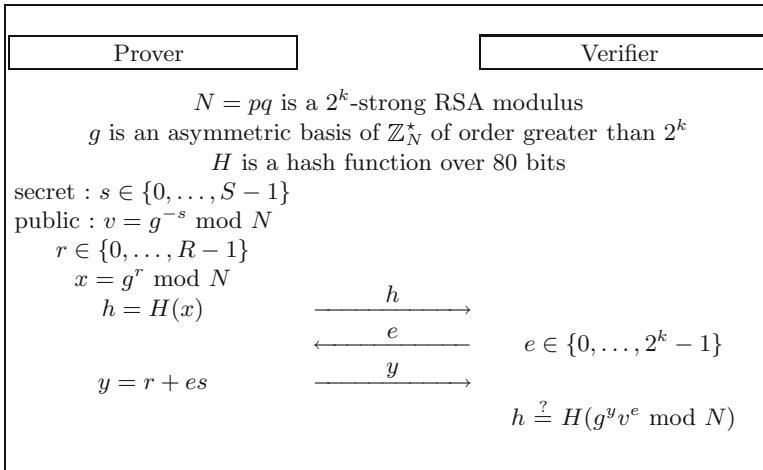


Fig. 2. Optimized Identification Scheme

Our Reduction. Using the witness-indistinguishable property, the reduction using a passive impersonation has the same complexity. On the other hand, an impersonation with probability ε greater than $2 \cdot 2^{-k}$ after ℓ active attacks within time T can be used to find two distinct answers for a same commitment within an expected time bound $4T/\varepsilon$, since no simulation is required.

Summary. An impersonation with probability ε , greater than 2×2^{-k} , after $\ell > 0$ active attacks can help to compute discrete logarithms within a time $2^k \ell T$ (thanks to the zero-knowledge property) or to factor N within a time $4T/\varepsilon$.

(thanks to the witness-indistinguishable property). This latter is much smaller than the former. Therefore, a high security level against active attacks can only be related to factorization: for example, with $k = 30$ and $\ell = 2^{40}$, the reduction cost is less than 2^{30} from factorization, against 2^{70} from discrete logarithm, which has no practical meaning!

Communication Load. With the presented proof, S can be chosen over very few hundreds of bits, provided $S \geq 2 \cdot \text{Ord}(g)$ (namely 160 bits to avoid baby steps–giant steps attacks [46]). Furthermore, the communication load can be optimized using the Girault and Stern’s technique [18] as it is presented on Figure 2. Indeed, a hash function that returns 80-bit digests requires 2^{64} computations to expect a 5-collision. Then, with a $k + 3$ -bit challenge, the security level remains 2^{-k} . Both remarks lead to very efficient and low-cost protocols (see Figure 2).

3.2 Signature

We can of course derive this identification scheme into a signature scheme Σ (see Figure 3), using a hash function to generate the random challenge. The security against existential forgeries under no-message attacks of the signature scheme is clear in the random oracle model [12, 26, 25, 37]. Because of the witness-

Initialisation
$N = pq$ is a 2^k -strong RSA modulus
g is an asymmetric basis of \mathbb{Z}_N^* of order greater than 2^k
Key Generation
secret key : $s \in \{0, \dots, S - 1\}$
public key: $v = g^{-s} \bmod N$
Signature of m
choose $r \in \{0, \dots, R - 1\}$ and compute $x = g^r \bmod N$
get $e = H(m, x)$ and compute $y = r + es$
$\Sigma(\mathbf{m}) = (\mathbf{e}, \mathbf{y})$
Verification of (m, e, y) : $e \stackrel{?}{=} H(m, g^y v^e \bmod N)$

Fig. 3. Signature Scheme

indistinguishable property, we need not any simulation for the security against adaptive chosen-message attacks. In fact, we can use a real signer with a secret key s_1 and use the forking lemma [37], or the ID reduction lemma [25], to extract a second one from the attacker. As previously seen for the identification scheme, if $S \geq 2 \cdot \text{Ord}(g)$, with high probability, we get the factorization of the modulus N .

Theorem 10. *With $S \geq 2 \cdot \text{Ord}(g)$, an existential forgery under an adaptive chosen-message attack of this scheme is harder than the factorization.*

3.3 Blind Signature

Now, we focus on a new blind signature scheme based on the previously seen problem. The construction of the blind signature is not straightforward because the initialization of this scheme requires the security parameters to be carefully chosen. However, the resulting scheme is very interesting from the bank point of view. Indeed, its computational load is minimal.

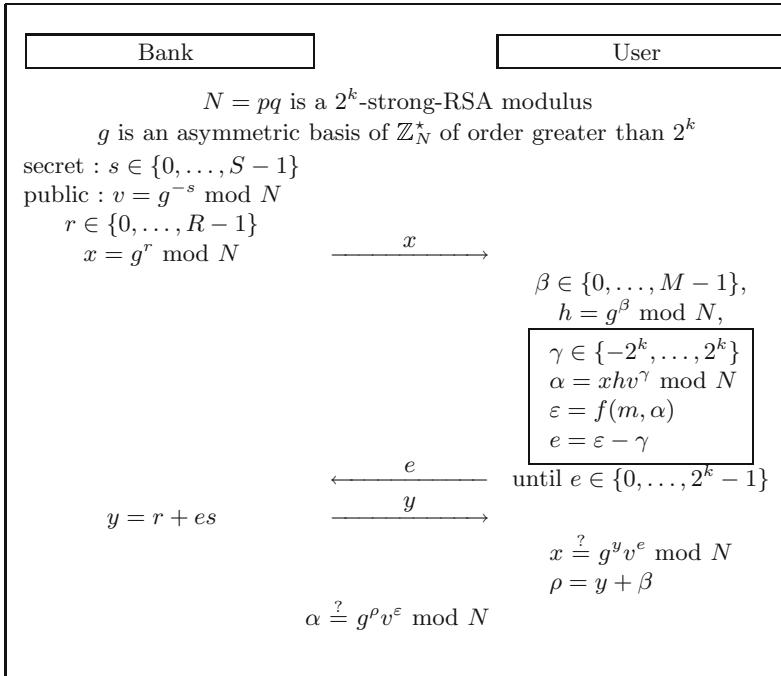


Fig. 4. Blind Signature Scheme

Presentation. Because of the witness-indistinguishable property seen above, we hope to get a blind signature scheme at least more secure than factorization. Let first present this scheme (see Figure 3.3), where k is the security parameter and k' the information leak parameter: we define $R = 2^{k+k'}S$ and $M = 2^{k+2k'}S$, where $S \geq 2 \cdot \text{Ord}(g)$ defines the range set of the secret key.

Security. First, we have to prove that this scheme is really blind, *i.e.* even a dishonest bank cannot link later a user and a message/signature pair. This is a fundamental property required by anonymous protocols (electronic cash, electronic voting). We want the bank not to be able to recognize a user even with the message and the signature.

Theorem 11. *This scheme is a statistically blind signature scheme.*

Proof. The output of this protocol is a signature which has been considered in the previous section and proven secure. Then, we only have to prove that the protocol is “blind”.

Let $(m, \alpha, \varepsilon, \rho)$ be a valid signature obtained from an execution of the “blind signature scheme” after one of both interactions (x_1, e_1, y_1) and (x_2, e_2, y_2) . Is it possible to know, with non-negligible advantage, from which one it comes? To know that, we have to study the following probabilities for $i = 1, 2$:

$$p_i(\alpha, \varepsilon, \rho) = \Pr_{\beta, \gamma}[\alpha = x_i g^\beta v^\gamma, \varepsilon = e_i + \gamma \text{ and } \rho = y_i + \beta \mid 0 \leq \varepsilon - \gamma \leq 2^k - 1].$$

For both values of i ,

$$\begin{aligned} p_i(\alpha, \varepsilon, \rho) &= \Pr_{\beta, \gamma}[\gamma = \varepsilon - e_i \text{ and } \beta = \rho - y_i \mid 0 \leq \varepsilon - \gamma \leq 2^k - 1] \\ &= \frac{\delta(0 \leq \rho - y_i \leq M - 1)}{M} \times 2^{-k} = \delta(y_i \leq \rho \leq M + y_i - 1)/2^k M. \end{aligned}$$

Then, the distance between both distributions is equal to

$$\Delta = \sum_{\varepsilon} \frac{2|y_2 - y_1|}{2^k M} \leq 2 \times \sum_{\varepsilon} \frac{2^k S(1 + 2^{k'})}{2^k M} \leq 2 \times \frac{1 + 2^{k'}}{2^{2k'}} \leq \frac{3}{2^{k'}}.$$

This distance is therefore negligible into the information leak parameter k' . \square

Now, we also claim the following security result:

Theorem 12. *A “one-more” forgery under a parallel attack against this blind signature scheme is harder than the factorization of N , in the random oracle model.*

Proof. The proof uses the same technique as [37], since it just takes advantage of the witness-indistinguishability. We therefore refer the reader to this paper. \square

4 Security and Efficiency

For practical purpose, it seems to be convenient to choose a 1024-bit modulus N and an asymmetric basis g of 160-bit long order. The information leak parameter k' can be fixed to 64, and the security parameter to 24 or 128 depending on the situation (see Figure 5).

Therefore, the security provably relies on the factorization of the 1024-bit modulus (which is assumed to be infeasible). In case of discovery of a new and efficient algorithm to factor large numbers, the security collapses at the same level as the Schnorr schemes: the discrete logarithms in subgroups of prime order (unproven for identification, but provable for signatures).

From the prover point of view, these protocols are very efficient. Indeed, if we only take in account the computation he has to do in real-time, it only consists

Scheme	Identification	Signature	Blind Signature
Modulus	$ N = 1024$ bits with $ p = q = 512$ bits		
$\text{Ord}(g)$		160 bits	
Security parameter	$k = 24$		$k = 128$
Information leak parameter		$k' = 64$	
$ S $ ($> \text{Ord}(g) $)		168 bits	
$ R $ ($= S + k + k'$)	256 bits		360 bits
$ M $ ($= S + k + 2k'$)			424 bits
Online Cost (prover)	$\text{Mult}(24,168) + \text{Add}(256,192)$		$\text{Mult}(128,168) + \text{Add}(360,296)$
Communication	360 bits (45 bytes)		
Signature Size		488 bits (61 bytes)	552 bits (69 bytes)

Fig. 5. Efficiency of the Proposed Schemes

of one multiplication and one addition over the natural integers \mathbb{N} . Furthermore, the used numbers are very small.

As one can remark, since the commitment can be precomputed, the prover has only one multiplication and one addition to perform during a proof (identification/signature/blind signature). For the recommended parameters, for a blind signature (the most costly scheme), the bank has just to multiply a 128-bit integer by a 168-bit one and to add the result to a 360-bit integer. The important gain versus the Schnorr schemes is the suppression of the modular reduction.

Then, at the cost of a little storage, the bank can blindly sign millions of messages per second, which can be required in a huge electronic cash application or electronic vote. Furthermore, thanks to the security result, parallel withdrawals can be performed securely.

5 Conclusion

In this paper, we have presented many schemes based on the composite discrete logarithm problem. From identification to blind signature, we have proven efficient schemes to be at least as secure as factorization.

The main contributions of this paper, vs. the Poupard and Stern's one [38], are the possible use of small secret keys and the security proof of the three-pass identification scheme even with large challenges, thanks to the witness-indistinguishability of the protocol. Both properties lead to the most efficient identification scheme known for the moment, with provable security (namely related to factorization). The non-interactive version, using a hash function [12], thus provides a very efficient signature scheme more secure than factorization, outputting very short signatures (approximatively 60 bytes).

Furthermore we provide a new blind signature scheme really efficient from the bank point of view. Indeed, the security result allows parallel withdrawals and the low computational load of the bank provides a very high rate. Therefore, this scheme is very well suited for very huge scale applications with thousands of users.

References

1. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993. [114](#)
2. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995. [114](#)
3. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996. [114](#)
4. S. A. Brands. An Efficient Off-Line Electronic Cash System Based on the Representation Problem. Technical Report CS-R9323, CWI, Amsterdam, 1993. [115](#)
5. S. A. Brands. Untraceable Off-Line Cash in Wallets with Observers. In *Crypto '93*, LNCS 773, pages 302–318. Springer-Verlag, Berlin, 1994. [115](#)
6. E. F. Brickell and K. S. McCurley. An Interactive Identification Scheme Based on Discrete Logarithms and Factoring. *Journal of Cryptology*, 5:29–39, 1992. [113](#), [115](#)
7. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218. ACM Press, New York, 1998. [114](#)
8. D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto '82*, pages 199–203. Plenum, New York, 1983. [115](#)
9. I. B. Damgård. Payment Systems and Credential Mechanisms with Provable Security against Abuse by Individuals. In *Crypto '88*, LNCS 403, pages 328–335. Springer-Verlag, Berlin, 1989. [115](#), [116](#)
10. P. de Rooij. On the Security of the Schnorr Scheme Using Preprocessing. In *Eurocrypt '91*, LNCS 547, pages 71–80. Springer-Verlag, Berlin, 1992. [114](#)
11. P. de Rooij. On Schnorr's Preprocessing for Digital Signature Schemes. *Journal of Cryptology*, 10:1–16, 1997. [114](#)
12. U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, 1:77–95, 1988. [113](#), [118](#), [122](#), [125](#)
13. U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *Proc. of the 22nd STOC*, pages 416–426. ACM Press, New York, 1990. [114](#), [115](#), [116](#), [117](#), [119](#)
14. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987. [113](#), [118](#)
15. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC '99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999. [114](#)
16. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999. [114](#)

17. M. Girault. Self-Certified Public Keys. In *Eurocrypt '91*, LNCS 547, pages 490–497. Springer-Verlag, Berlin, 1992. [113](#), [115](#), [116](#), [118](#)
18. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto '94*, LNCS 839, pages 202–215. Springer-Verlag, Berlin, 1994. [122](#)
19. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304. ACM Press, New York, 1985. [113](#)
20. L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In *Eurocrypt '88*, LNCS 330, pages 123–128. Springer-Verlag, Berlin, 1988. [113](#), [114](#)
21. L. C. Guillou and J.-J. Quisquater. A “Paradoxal” Identity-Based Signature Scheme Resulting from Zero-Knowledge. In *Crypto '88*, LNCS 403, pages 216–231. Springer-Verlag, Berlin, 1989. [113](#)
22. A. Juels, M. Luby, and R. Ostrovsky. Security of Blind Digital Signatures. In *Crypto '97*, LNCS 1294, pages 150–164. Springer-Verlag, Berlin, 1997. [115](#), [116](#)
23. SET Secure Electronic Transaction LLC. SET Secure Electronic Transaction Specification – Book 3: Formal Protocol Definition, may 1997.
Available from <http://www.setco.org/>. [114](#)
24. G. Miller. Riemann’s Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13:300–317, 1976. [119](#)
25. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Crypto '98*, LNCS 1462, pages 354–369. Springer-Verlag, Berlin, 1998. [114](#), [115](#), [118](#), [121](#), [122](#)
26. T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *Crypto '92*, LNCS 740, pages 31–53. Springer-Verlag, Berlin, 1992. [114](#), [115](#), [122](#)
27. T. Okamoto, S. Uchiyama, and E. Fujisaki. EPOC: Efficient Probabilistic Public-Key Encryption. Submission to IEEE P1363a. November 1998.
Available from <http://grouper.ieee.org/groups/1363/addendum.html>. [114](#)
28. H. Ong and C.P. Schnorr. Fast Signature Generation with a Fiat-Shamir-Like Scheme. In *Eurocrypt '90*, LNCS 473, pages 432–440. Springer-Verlag, Berlin, 1991. [113](#), [116](#)
29. P. Paillier and D. Pointcheval. Efficient Public-Key Cryptosystems Provably Secure against Active Adversaries. In *Asiacrypt '99*, LNCS 1716. Springer-Verlag, Berlin, 1999. [114](#)
30. B. Pfitzmann and M. Waidner. How to Break and Repair a “Provably Secure” Untraceable Payment System. In *Crypto '91*, LNCS 576, pages 338–350. Springer-Verlag, Berlin, 1992. [115](#), [116](#)
31. D. Pointcheval. A New Identification Scheme Based on the Perceptrons Problem. In *Eurocrypt '95*, LNCS 921, pages 319–328. Springer-Verlag, Berlin, 1995. [113](#)
32. D. Pointcheval. New Public Key Cryptosystems based on the Dependent-RSA Problems. In *Eurocrypt '99*, LNCS 1592, pages 239–254. Springer-Verlag, Berlin, 1999. [114](#)
33. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '00*, LNCS. Springer-Verlag, Berlin, 2000. [114](#)
34. D. Pointcheval and J. Stern. Provably Secure Blind Signature Schemes. In *Asiacrypt '96*, LNCS 1163, pages 252–265. Springer-Verlag, Berlin, 1996. [114](#), [115](#)
35. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, Berlin, 1996. [114](#)

36. D. Pointcheval and J. Stern. New Blind Signatures Equivalent to Factorization. In *Proc. of the 4th CCS*, pages 92–99. ACM Press, New York, 1997. [114](#), [116](#)
37. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 1999.
Available from <http://www.di.ens.fr/~pointche>. [114](#), [115](#), [116](#), [117](#), [118](#), [121](#), [122](#), [124](#)
38. G. Poupard and J. Stern. Security Analysis of a Practical “on the fly” Authentication and Signature Generation. In *Eurocrypt ’98*, LNCS 1403, pages 422–436. Springer-Verlag, Berlin, 1998. [114](#), [115](#), [116](#), [119](#), [121](#), [125](#)
39. G. Poupard and J. Stern. On The Fly Signatures based on Factoring. In *Proc. of the 6th CCS*. ACM Press, New York, 1999. [115](#)
40. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *PKC ’00*, LNCS. Springer-Verlag, Berlin, 2000. [115](#)
41. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. [114](#), [115](#)
42. RSA Data Security, Inc. Public Key Cryptography Standards – PKCS.
Available from <http://www.rsa.com/rsalabs/pubs/PKCS/>. [114](#)
43. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto ’89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990. [113](#), [114](#), [116](#), [120](#)
44. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991. [114](#), [120](#)
45. A. Shamir. An Efficient Identification Scheme Based on Permuted Kernels. In *Crypto ’89*, LNCS 435, pages 606–609. Springer-Verlag, Berlin, 1990. [113](#)
46. D. Shanks. Class number, a theory of factorization, and genera. In *Proceedings of the symposium on Pure Mathematics*, volume 20, pages 415–440. AMS, 1971. [122](#)
47. V. Shoup. On The Security of a Practical Identification Scheme. In *Eurocrypt ’96*, LNCS 1070, pages 344–353. Springer-Verlag, Berlin, 1996. [116](#)
48. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt ’98*, LNCS 1403, pages 1–16. Springer-Verlag, Berlin, 1998. [114](#)
49. J. Stern. A New Identification Scheme Based on Syndrome Decoding. In *Crypto ’93*, LNCS 773, pages 13–21. Springer-Verlag, Berlin, 1994. [113](#)
50. J. Stern. Designing Identification Schemes with Keys of Short Size. In *Crypto ’94*, LNCS 839, pages 164–173. Springer-Verlag, Berlin, 1994. [113](#)

Chosen-Ciphertext Security for Any One-Way Cryptosystem

David Pointcheval

Dépt d’Informatique, ENS – CNRS, 45 rue d’Ulm, 75230 Paris Cedex 05, France.
David.Pointcheval@ens.fr – URL: <http://www.di.ens.fr/~pointche>.

Abstract. For two years, public key encryption has become an essential topic in cryptography, namely with security against chosen-ciphertext attacks. This paper presents a generic technique to make a highly secure cryptosystem from any partially trapdoor one-way function, in the random oracle model. More concretely, any suitable problem providing a one-way cryptosystem can be efficiently derived into a chosen-ciphertext secure encryption scheme. Indeed, the overhead only consists of two hashing and a XOR. As application, we provide the most efficient El Gamal encryption variant, therefore secure relative to the computational Diffie-Hellman problem. Furthermore, we present the first scheme whose security is relative to the factorization of large integers, with a perfect reduction (factorization is performed within the same time and with identical probability of success as the security break).

1 Introduction

1.1 Background

In 1976, Diffie and Hellman [13] proposed the concept of public-key cryptography, and namely of public-key encryption using trapdoor one-way functions. However, despite research efforts of many cryptographers, very few practical cryptosystems have been proposed.

In the mean time, a lot of security notions have been proposed, from one-wayness, where the attacker tries to recover the whole plaintext given the ciphertext and only public data, to non-malleability under chosen-ciphertext attacks [14,32,6], where the attacker tries to derive a new ciphertext, whose plaintext is meaningfully related to the original one, with access to the decryption algorithm.

More recently, a general study of security notions [2] has been driven, leading to the conclusion that all those notions are equivalent under chosen-ciphertext attacks, which we regroup under the name of “chosen-ciphertext security”. It therefore represents the most powerful of the practical security notions, just under the theoretical notion of plaintext-awareness, only defined in the random oracle model (see [4,2] for more details). For a long time, one-wayness or even just heuristic security of an encryption scheme have been considered enough.

Semantic security [18]/non-malleability [14] were just seen as theoretical properties, for people from the complexity theory. But after the numerous practical attacks, namely the recent Bleichenbacher's [9], Coron-Naccache-Stern's and Coppersmith-Halevi-Jutla's [11,10] ones, provable security has been realized to be of important interest for many applications. Therefore, chosen-ciphertext security has become a requirement for encryption schemes.

1.2 Related Work

The reason of this small number of candidates as secure cryptosystems relies on the fact that hardly satisfied properties are required. And even with a well-suited problem, a cryptosystem is not easily derived and does not necessarily lead to an efficient resulting scheme.

Before 1994, only theoretical and impractical schemes were proposed [22,32]. Then Bellare and Rogaway [4] designed a generic padding, called Optimal Asymmetric Encryption Padding, to make a chosen-ciphertext secure cryptosystem from any trapdoor one-way permutation. However, such permutations are very rare: indeed, RSA [33] is the only one (with the recently proposed, at last PKC '99, by Paillier [26], but also based on the RSA assumption). Therefore OAEP–RSA has been a long time the only practical cryptosystem secure against chosen-ciphertext attacks. Then it has been incorporated in SET, the Secure Electronic Transaction system [19] proposed by VISA and Master Card, and has become the new RSA encryption standard PKCS #1 v2.0 [34].

The last few years, many new schemes have been proposed with proven security relative to decisional problems:

- the decisional Diffie–Hellman problem [13]: at PKC '98, Tsiounis–Yung [38] proposed the first El Gamal [15] based cryptosystem, but using an unproven assumption about the unforgeability of Schnorr's Signatures [35]. The same year, Shoup–Gennaro [37] and Cramer–Shoup [12] proposed other variants, the latter is even secure in the standard model.
- the decisional dependent–RSA problem: the author of this work [29,30] proposed schemes based on a new problem, called the dependent–RSA problem. Some variants have also been proven chosen-ciphertext secure relative to RSA, which became the first alternative to the OAEP–RSA.
- the higher residuosity [7,8]: Paillier–Pointcheval [28] proposed a variant of the Paillier's scheme [27], whose main interest is the efficiency of the decryption process.

However, those schemes came one by one, with new and specific intricate proofs of security for each. Last year, at PKC '99, Fujisaki–Okamoto [16] proposed a second generic transformation. This very simple transformation enhances the security of any public-key encryption scheme in the random oracle model: from a semantically secure scheme (against just chosen-plaintext attacks), it makes a chosen-ciphertext secure scheme. It may be applied to many, more or less recently proposed, schemes also based on above decisional problems [18,15,21,24]. In other

words, any trapdoor decisional problem can be derived into chosen-ciphertext secure schemes.

However, decisional problems generally rely on strong assumptions, whereas one-way schemes are based on weaker ones, such as the classical computational problems: RSA [33], Diffie–Hellman [15], factorization [24], classes of residuosity/partial discrete logarithm [27]), etc. Indeed, computational problems are clearly more difficult to solve and are moreover more numerous than decisional ones.

At last crypto, Fujisaki–Okamoto [17] improved their generic transformation, to make security related to the computational problem instead of just the decisional one.

1.3 Outline of the Paper

The present research, independently driven from the Fujisaki–Okamoto [17] work, reaches a similar result but with more efficient reductions: it presents the most efficient and general transformation. Indeed, it allows to make chosen-ciphertext secure schemes from any one-way encryption scheme: for any one-way function, if a trapdoor allows to get back a part of the preimage, one can base a chosen-ciphertext secure encryption scheme on the relying computational problem.

More concretely, from any one-way encryption scheme (which is the weakest requirement one can make about an encryption scheme) and just two more hashing, one can make a highly secure cryptosystem relying only on the same assumption as the one-wayness of the original scheme, which is generally a really difficult computational problem (at least more difficult than just decisional ones).

We then apply this generic transformation to many well-known one-way functions to provide the best schemes of their families: the most efficient scheme based on the computational Diffie–Hellman problem and the first scheme as secure as factorization.

2 Security Notions for Public Key Encryption

2.1 Definitions

The first common security notion that one would like an encryption scheme to satisfy is the *one-wayness*: with just public data, an attacker can not get back the whole plaintext of a given ciphertext. This notion was satisfied by the RSA cryptosystem [33], relative to the RSA assumption, and by the El Gamal encryption scheme [15], relative to the computational Diffie–Hellman problem. Many applications require more from an encryption scheme, namely

- semantic security (a.k.a. *polynomial security/indistinguishability of encryptions* [18]): if the attacker has some information about the plaintext, for example that it is either “yes” or “no” to a crucial query, she should not

learn more with the view of the ciphertext. It is computationally impossible to distinguish between two messages which one has been encrypted. This implies encryption schemes to be probabilistic, such as the El Gamal scheme [15], but not like RSA [33].

- non-malleability [14]: for the problem of encrypted bids, an attacker may just want to under-bid a ciphertext of an unknown amount, without learning anything about this amount or her own proposition. This property has been formally defined under the notion of *non-malleability* [14,6]: from a given ciphertext any attacker can not derive a new ciphertext in such a way that the plaintexts underlying the two ciphertexts are meaningfully related.

On the other hand, an attacker can play many kinds of attacks: she may just have access to public data, and then encrypt any plaintext of her choice (*chosen-plaintext attacks*) or moreover query the decryption algorithm (*adaptively/non-adaptively chosen-ciphertext attacks* [22,32]).

A general study of these security notions has been recently driven [2], we therefore refer the reader to this paper for more details, concluding with a complete hierarchy. More precisely, semantic security and non-malleability are equivalent in the adaptively chosen-ciphertext scenario, which defines the strongest practical security notion. In what follows, we call it “*chosen-ciphertext security*”.

2.2 Model of Security

For the last few years, after the numerous attacks against unprovable schemes, provable security has been realized to be of greatest interest. Such a proof is led in the complexity theory setting: one tries to polynomially reduce a well-established difficult problem to an attack. Therefore, an efficient attacker would help to solve the difficult problem: this leads to a contradiction.

Very few schemes have been proven using only such polynomial reductions, without any other assumption. Furthermore, they hardly reach efficiency. The last years, the so-called “random oracle model” [3] has boosted researches, providing an interesting tool in proving the security of very efficient schemes. Indeed, this model, where some concrete cryptographic objects are idealized, namely the hash functions which are assumed to be really random ones, helped to provide security proofs for many encryption schemes [3,4,37,29,25,16,17,28] and digital signature schemes [5,23,31].

3 The New Construction

Our new construction can be applied from any partially trapdoor one-way injective function, and not just from fully trapdoor one-way permutations [4] or already semantically secure encryption schemes [16]. This long sentence “partially trapdoor one-way injective function” is nothing else than a classical encryption scheme which is secure in the weakest sense, just one-way against chosen-plaintext attacks.

Then, in order to formalize notations, we first define this notion of partially trapdoor one-way functions, which informally characterizes one-way functions for which a trapdoor allows a partial recovery of a preimage.

3.1 Partially Trapdoor One-Way Function

Let us consider any function f , from the product space $\mathcal{X} \times \mathcal{Y}$ into \mathcal{Z} .

Definition 1 (One-Way). *The function f is said to be one-way if, for any given $z = f(x, y)$, it is computationally impossible to get back the couple (x, y) . More formally, for any polynomial time adversary \mathcal{A} , its success $\text{Succ}_{\mathcal{A}}$, defined by $\text{Succ}_{\mathcal{A}} = \Pr_{x,y}[f(\mathcal{A}(f(x, y))) = f(x, y)]$, is negligible.*

Whereas we will use the above denomination in the rest of the paper, our result will deal with an even weaker notion of one-wayness, where an adversary should not only have to invert with non-negligible probability of success, but she should also have to rarely output wrong solutions. Of course, she is allowed to output “Reject” when she can not solve the problem.

Definition 2 (Weakly One-Way). *The function f is said to be weakly one-way if, it is either one-way or, for any polynomial time adversary \mathcal{A} , its error probability, defined by*

$$\text{Err}_{\mathcal{A}} = \Pr_{x,y}[f(u, v) \neq f(x, y) \mid \mathcal{A}(f(x, y)) \neq \text{“Reject”} \wedge (u, v) = \mathcal{A}(f(x, y))],$$

is non-negligible.

It is a weaker assumption about a problem. Indeed, an adversary may be able to return a correct answer half the time, and therefore break “one-wayness”. But the other half of answers can be junk, which can not be detected if the decisional problem is also difficult (such as the Diffie-Hellman problem [13], the Residuosity problem [7,8,24,21,27], etc).

Definition 3 (Trapdoor). *A (weakly) one-way function is said to be trapdoor, if for some extra information (the trapdoor), for any given $z \in f(\mathcal{X} \times \mathcal{Y})$, it is easily possible to get back a couple (x, y) such that $f(x, y) = z$. Whereas it was computationally impossible without the trapdoor.*

As already remarked, such functions which also need to be injective for cryptographic use are very rare (just RSA, and some few related ones), but they are required to apply OAEP [4]. This relativizes the practical impact of the OAEP-transformation.

However, for encryption purpose, it is not required to get back the whole preimage of z , it is the reason why we define the new *partially trapdoor one-way* notion which is more common (see El Gamal [15], or more recently Okamoto-Uchiyama [24], Naccache-Stern [21] and Paillier [27]).

Definition 4 (Partially Trapdoor One-Way). *The function f is said to be partially trapdoor one-way if,*

- for any given $z = f(x, y)$, it is computationally impossible to get back an available x . Such an x is called a partial preimage of z .
- More formally, for any polynomial time adversary \mathcal{A} , its success, defined by $\text{Succ}_{\mathcal{A}} = \Pr_{x,y}[\exists y', f(x', y') = f(x, y) \mid x' = \mathcal{A}(f(x, y))]$, is negligible. It is one-way even for just finding partial-preimage, thus partial one-wayness.
- for some extra information, for any given $z \in f(\mathcal{X} \times \mathcal{Y})$, it is easily possible to get back an x , such that there exists a y which satisfies $f(x, y) = z$. The trapdoor does not allow a total inversion, but just a partial one, it is thus called a partial trapdoor.

Definition 5 (Partially Trapdoor Weakly One-Way). The function f is said to be partially trapdoor weakly one-way if it is partially trapdoor one-way but furthermore, for any polynomial time adversary \mathcal{A} , either its success $\text{Succ}_{\mathcal{A}}$ is negligible or its error probability, defined by

$$\text{Err}_{\mathcal{A}} = \Pr_{x,y}[\forall y', f(x', y') \neq f(x, y) \mid \mathcal{A}(f(x, y)) \neq \text{"Reject"} \wedge (x', y') = \mathcal{A}(f(x, y))],$$

is non-negligible. It is weakly one-way even for just finding partial-preimage, thus partial one-wayness.

Such partially trapdoor (weakly) one-way functions, moreover *injective*, are of common use in many cryptosystems, however they usually only provide the “one-wayness” of the encryption scheme, which is a very weak property for a cryptosystem. Whereas even semantic security against chosen-plaintext attacks relies on much stronger assumptions or is simply not provable/achieved. Let us briefly recall some of them.

3.2 Some Partially Trapdoor One-Way Injective Functions

The Diffie-Hellman Problem. The most popular encryption scheme based on a partially trapdoor one-way function is the El Gamal cryptosystem [15] based on the Diffie-Hellman distribution key problem [13].

- The Computational Diffie-Hellman Problem: given g , g^a and g^b in a group G , compute g^{ab} .
- The Decisional Diffie-Hellman Problem: given g , g^a , g^b and g^c in a group G , decide whether $g^c = g^{ab}$ or not.
- The El Gamal encryption scheme: given a message m (theoretically in G), a ciphertext is a pair $(g^a, y^a \cdot m)$, where $y = g^b$ is the public key, while b is kept secret.

Computing g^{ab} just given g^a and g^b is assumed impossible (Computational Diffie-Hellman Problem), whereas given b , it only consists of an exponentiation. Then the partially trapdoor one-way function is the following, where $y = g^b$, and q the order of g :

$$\begin{array}{ll} f : G \times \mathbb{Z}_q \longrightarrow G \times G & g : G \times G \longrightarrow G \\ (m, a) \longmapsto (g^a, y^a \cdot m) & (x, z) \longmapsto m = z/x^b \end{array}$$

The one-wayness of the El Gamal encryption scheme clearly relies on the *partial* one-wayness of this function, without the knowledge of b : the Computational Diffie-Hellman Problem, which is almost as difficult as the discrete logarithm problem [20]. However, semantic security requires a much stronger assumption, the Decisional Diffie-Hellman one.

The Partial Discrete Logarithm Problem. More recently, at Crypto '98, Okamoto-Uchiyama [24], at ACM CCS '98, Naccache-Stern [21] and, at Eurocrypt '99, Paillier [27] proposed new encryption schemes based on trapdoor discrete logarithms. More precisely, a trapdoor (the factorization of the composite modulus) allows to partially compute discrete logarithms. The encryption process puts the message to be encrypted in this recoverable part. The one-wayness of those schemes relies on the factorization, the higher residues and the partial discrete logarithms respectively. However, the semantic security (even against chosen-plaintext attacks) relies on higher residues [7,8], a weaker problem than factorization and even RSA [27].

The aim of this work is to provide a generic transformation to make any encryption scheme, whose one-wayness is provable, semantically secure even against adaptively chosen-ciphertext attacks, adding just the random oracle assumption.

3.3 Generic Construction

Let us consider such a partially trapdoor one-way injective function f , from the product space $\mathcal{X} \times \mathcal{Y}$ into \mathcal{Z} , and we denote by g its partial invert:

$$\begin{array}{ll} f : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{Z} & g : \mathcal{Z} \longrightarrow \mathcal{X} \\ (x, y) \longmapsto z & z \longmapsto x \quad \text{s.t. } \exists y \in \mathcal{Y}, z = f(x, y) \end{array}$$

We furthermore need two functions, a hash function H and a generator function G , both assumed to be ideal random functions [3], where k is a security parameter:

$$H : \{0, 1\}^k \rightarrow \mathcal{Y} \quad G : \mathcal{X} \rightarrow \{0, 1\}^k.$$

The names, “hash” and “generator” functions, come from the fact that, in practice, \mathcal{X} and \mathcal{Y} will be of similar size, but maybe smaller than $\{0, 1\}^k$. The cryptosystem is designed in Figure 1, with $k = k_0 + k_1$, where k_0 and k_1 denote the lengths of the messages to be encrypted and the error-parameter respectively. Moreover, $[M]_{k_0}$ denotes the truncation of the bit-string M to its k_0 left bits.

Concerning this scheme, called $\mathcal{E}nc_f$, we show that, under some assumptions about the function f , an attacker against semantic security under an adaptively chosen-ciphertext attack can be used to efficiently simulate g , and thus partially invert the one-way function f without the trapdoor information, which is computationally impossible under the *partially trapdoor one-way assumption* for the function f .

In what follows, X and Y denote the sizes of \mathcal{X} and \mathcal{Y} respectively, whereas q_G , q_H and q_D denote the numbers of queries asked to the random oracles G and H and to the decryption oracle D , respectively.

Encryption of $m \in \mathcal{M} = \{0, 1\}^{k_0} \rightarrow (\mathbf{a}, \mathbf{b})$
$r \in \mathcal{X}$ and $s \in \{0, 1\}^{k_1}$ are randomly chosen
$a = f(r, H(m\ s))$
$b = (m\ s) \oplus G(r)$
————— (a, b) is the ciphertext
Decryption of (\mathbf{a}, \mathbf{b})
Given $a \in \mathcal{Z}$ and $b \in \{0, 1\}^k$
$r = g(a)$
$M = b \oplus G(r)$
if $a = f(r, H(M))$
————— $m = [M]_{k_0}$ is the plaintext
otherwise, “Reject: invalid ciphertext”

Fig. 1. Our Generic Construction $\mathcal{E}nc_f$

Lemma 6. *Let us consider an attacker \mathcal{A} against the semantic security of $\mathcal{E}nc_f$ in a chosen-plaintext scenario. If we denote by ε the advantage of this attacker, one can design an algorithm \mathcal{B} that outputs, for any given z , a set \mathcal{S} of values such that a partial preimage of z is in \mathcal{S} with probability greater than $\varepsilon - q_H/2^{k_1}$.*

Proof. Let us consider an adversary $\mathcal{A} = (A_1, A_2)$ against the semantic security of this scheme, where A_1 denotes the “find”-stage and A_2 the “guess”-stage. We then use this adversary to construct a machine \mathcal{B} able to output candidates as partial preimages of f . Let z be a given value in \mathcal{Z} for which we want to find the partial preimage in \mathcal{X} . Our machine \mathcal{B} works as follows:

- It first runs the attacker A_1 where any query to the oracles G and H are intercepted. For any new query q asked to the oracle H , \mathcal{B} chooses a random H_q in \mathcal{Y} and outputs it as the value $H(q)$. The same way, for any new query q asked to the oracle G , \mathcal{B} chooses a random G_q in $\{0, 1\}^k$ and outputs it as the value $G(q)$. The attacker A_1 finally outputs two messages m_0 and m_1 . Our machine \mathcal{B} chooses a random bit c and a random string $b \in_R \{0, 1\}^k$, then it defines $a = z$ and outputs (a, b) as an encryption of m_c .
- The attacker A_2 is fed with this ciphertext (a, b) , and queries to oracles are again intercepted and answered as above.
- When the attacker returns its answer d , our machine \mathcal{B} returns the set \mathcal{S} of all the queries asked to the oracle G during the whole attack.

Now, let us assume that $z = f(x, y)$ for some (x, y) . Because of injectivity, if such a pair exists, it is unique. We consider the game presented in Figure 2, where some values of the oracle are defined, if they have not already been. We define the following events:

- AskG, the query x is asked to G ;
- AskH, a query $m\|s$ is asked to H , for some message $m \in \mathcal{M}$, but the specific value s chosen at the beginning of the game.

We say that the attacker wins the game if some of both events occur or if, at the end, the value d returned by A_2 is equal to c . Then the advantage of the attacker is defined by $\text{Adv} = 2\Pr[\text{wins}] - 1$.

For a given $z = f(x, y)$

$$a \stackrel{\text{def}}{=} z, c \xleftarrow{R} \{0, 1\}, s \xleftarrow{R} \{0, 1\}^{k_1}, b \xleftarrow{R} \{0, 1\}^k$$

all the calls to G and H are intercepted

if AskG or AskH the game stops and the attacker wins

$$m_0, m_1 \leftarrow A_1^{G, H}(pk)$$

$$H(m_c \| s) \stackrel{\text{def}}{=} y \text{ and } G(x) \stackrel{\text{def}}{=} b \oplus m_c \| s$$

$$d \leftarrow A_2^{G, H}(a, b)$$

if $d = c$ the attacker wins

Fig. 2. The Game

With a random simulation of G and H , as described above, it is clear that this game perfectly simulates the real life excepted the unlikely case where x or $m_c \| s$ have already been asked to G or H respectively during the find stage (before their assignment). But this case makes the attacker to win in our game, then $\text{Adv} \geq \text{Adv}_{\mathcal{A}} = \varepsilon$. However, since no advantage can be gained by the adversary without AskG nor AskH, by splitting the game in two cases, depending on both events AskG and AskH, one obtains that $\Pr[\text{wins}] \leq \frac{1}{2} \times (1 + \Pr[\text{AskG} \vee \text{AskH}])$. Finally, this leads to $\Pr[\text{AskG} \vee \text{AskH}] \geq \varepsilon$.

Another remark that one can do is that AskH is very unlikely without AskG since it is the only way to gain information about s . More precisely,

$$\Pr[\text{AskH} \mid \neg \text{AskG}] \leq \frac{q_H}{2^{k_1}}.$$

Finally, one can conclude that $\varepsilon \leq \Pr[\text{AskG}] + \Pr[\text{AskH} \mid \neg \text{AskG}]$, and therefore

$$\Pr[\text{AskG}] \geq \varepsilon - \frac{q_H}{2^{k_1}}.$$

This means that with probability greater than $\varepsilon - q_H/2^{k_1}$, x lies in the set \mathcal{S} of queries asked to the oracle G . \square

Thanks to an easy simulation (a plaintext-extractor [4]), one can state the following result.

Theorem 1. *Let us consider an attacker \mathcal{A} against the semantic security of Enc_f in a chosen-ciphertext scenario. If we denote by ε the advantage of this*

attacker, one can design an algorithm \mathcal{B} that outputs, for any given z , a set \mathcal{S} of values such that a partial preimage of z is in \mathcal{S} with probability greater than

$$\varepsilon = \frac{q_H + q_D}{2^{k_1}} - \frac{q_D}{Y}.$$

Proof. Since we have the semantic security against chosen-plaintext attacks, we just have to provide a plaintext-extractor [4], to prove the plaintext-awareness of this scheme which implies security against chosen-ciphertext attacks [2]. The plaintext-extractor is a simulator of the decryption oracle. For a given cipher (a, b) , it works as follows:

- The simulator \mathcal{S} considers all the queries asked to G and H , (r, G_r) and (q, H_q) , and checks if for some pair (r, q) , both equalities $a = f(r, H_q)$ and $b = q \oplus G_r$ hold.
- At most one pair may satisfy both equalities (since f is an injection, otherwise it is not an encryption scheme). If one exists, q provides $m = [q]_{k_0}$ and s , and m is returned. Otherwise, the ciphertext is considered as an invalid one, and therefore rejected.

With this simulation, it is clear that only valid ciphertexts will be decrypted. But will all valid ciphertexts be decrypted? Definitely not, since a valid ciphertext can be produced without asking queries to both G and H . But since f is an injection, at most one value for $H(m||s)$ can be accepted: y , if $a = f(x, y)$.

$$\begin{aligned} \Pr[\text{valid} \mid \neg(\text{AskG} \wedge \text{AskH})] &= \Pr[\text{valid} \wedge (\neg\text{AskG} \vee \neg\text{AskH})] / \Pr[\neg\text{AskG} \vee \neg\text{AskH}] \\ &\leq \Pr[\text{valid} \wedge \neg\text{AskH}] / \Pr[\neg\text{AskH}] + \Pr[\text{valid} \wedge \neg\text{AskG} \wedge \text{AskH}] / \Pr[\neg\text{AskG}] \\ &\leq \Pr[\text{valid} \mid \neg\text{AskH}] + \Pr[\text{valid} \mid \neg\text{AskG}] \leq \frac{1}{Y} + \frac{1}{2^{k_1}} \end{aligned}$$

Finally, the probability of wrong decryption (rejection of valid ciphertext) is upper-bounded by $1/Y + 1/2^{k_1}$. Therefore, the probability to get no wrong decryption during the attack is lower-bounded by

$$\left(1 - \frac{1}{Y} - \frac{1}{2^{k_1}}\right)^{q_D} \geq 1 - \frac{q_D}{Y} - \frac{q_D}{2^{k_1}},$$

which concludes the proof. \square

Remark 1. As one can remark, if \mathcal{Y} is too small, which can even be empty in the case of a fully trapdoor function, and therefore Y not exponentially large, above result is meaningless. However, one can easily extend \mathcal{Y} : let f be a partially trapdoor one-way function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$, for any ℓ one defines $\mathcal{Y}_\ell = \mathcal{Y} \times \{0, 1\}^\ell$ and $\mathcal{Z}_\ell = \mathcal{Z} \times \{0, 1\}^\ell$ as well as

$$\begin{aligned} f_\ell : \mathcal{X} \times \mathcal{Y}_\ell &\longrightarrow \mathcal{Z}_\ell \\ (x, (y||r)) &\longmapsto f(x, y)||r \end{aligned}$$

Then, with no computational extra cost, one exponentially increases the size of the set \mathcal{Y} : $Y_\ell = Y \cdot 2^\ell$. However, in many cases, such extensions are not required.

Depending on the kind of problem, even the **weak one-wayness** can be really broken with various efficiency. However, with no particular extra property, randomly choosing x in the set \mathcal{S} , one can just break **one-wayness**, and then state the “General Case Theorem”.

Theorem 9 (The General Case). *Let us consider an attacker \mathcal{A} against the semantic security of Enc_f in a chosen-ciphertext scenario. If we denote by ε the advantage of this attacker, one can design an algorithm \mathcal{B} that returns, for any given z , a candidate as partial preimage of z by f which is correct with probability greater than*

$$\frac{1}{q_G} \times \left(\varepsilon - \frac{q_H + q_D}{2^{k_1}} - \frac{q_D}{Y} \right).$$

However, mathematical problems used in cryptography very often also satisfy a *strong* random self-reducible property (RSA, Diffie-Hellman, etc): from any instance can be derived a random instance whose solution easily provides a solution to the initial instance, using a *strong ring*-homomorphic reduction, where a *strong ring* is a ring whose cardinality only possesses large prime factors. This is usually the case, since the ring is generally, either a large prime field (Diffie-Hellman problem) or a \mathbb{Z}_n -ring, where n is an RSA-modulus ($n = pq$, RSA, residuosity, partial discrete logarithm [27]) or at least difficult to factor ($n = p^2q$ [24]).

Such problems, as any random self-reducible problem, have the particularity to be uniformly difficult (or easy), there is no worst case nor best case but just average ones. For a *strong* random self-reducible problem, one can then state an improved result. It is derived from a generalization of the Shoup’s theorem [36] about the faulty Diffie-Hellman oracles.

Lemma 10. *Let us consider a (k, δ) -oracle \mathcal{A} for a strong random self-reducible problem, which returns a list of k candidates that actually contains the solution with probability greater than $\delta > 7/8$. We can construct a probabilistic algorithm that breaks the **weak one-wayness** of the problem with the following properties. For given $\ell \in \mathbb{N}$, the algorithm makes 24ℓ queries to the oracle \mathcal{A} and performs $\mathcal{O}(\ell \cdot k)$ self-reductions. For all inputs, the output is correct with probability at least $1 - 2^{-\ell}$.*

Proof. With a strong random self-reducible partially one-way problem denoted by $f : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{Z}$, the structure $(\mathcal{X}, +, *)$ is a strong ring, of size X , and there exists a function $R : \mathcal{Z} \times \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{Z}$, which is invertible for any fixed second and third parameters. The function R maps any instance in \mathcal{Z} into a random one, using random elements r_+, r_* in \mathcal{X} , in such a way that the solution of the resulting instance x' is related with the solution of the initial instance x by $x' = x * r_* + r_+$. We assume that p , the smallest prime factor of X , is large enough and namely that $k^2 < p/8$.

For a given instance $z \in \mathcal{Z}$, one runs twice the oracle \mathcal{A} : once to find candidates, and a second time to check which one is correct. First, with input z , whose solution is x , it gets (x_1, \dots, x_k) . Then, with input $z' = R(z, r_1, r_2)$, for

some random r_1, r_2 , whose solution is $x' = x * r_1 + r_2$, it gets (x'_1, \dots, x'_k) . If a right solution is found in both lists, then $x = x_i$ and $x * r_1 + r_2 = x'_j$ for some (i, j) . Therefore, for some pair (i, j) ,

$$x'_j = x_i * r_1 + r_2. \quad (1)$$

Now, let us assume that, $x_i \neq x$, then the probability that above equality (1) holds is at most the conditional probability that for random elements r_1 and r_2 , $x'_j = x_i * r_1 + r_2$, given $x' = x * r_1 + r_2$. This is equal to the probability that for fixed x' and random r , $x'_j = (x_i - x) * r + x'$. Thanks to Shoup's Lemma 1, in [36], one knows that this latter is at most $1/p$.

Then, our algorithm either outputs x_i , if exactly one pair (i, j) satisfies equality (1), or reports failure. And therefore, three exclusive events may happen: (F) failure, (I) incorrect output, (C) correct answer. $\Pr[F] + \Pr[I]$ is upper bounded by the probability that one of the lists does not contain the correct output or that an extraneous relation (1) holds. This occurs with probability bounded by $1/8 + 1/8 + k^2/p \leq 3/8$. However, incorrect output can just occur when at least one of lists does not contain the correct output: $\Pr[I] \leq 1/8 + 1/8 = 1/4$. Then it follows that $\Pr[C] \geq 1 - 3/8 > 5/8$. Therefore

$$(\Pr[C] + \Pr[I]) / \Pr[I] = 1 + \Pr[C] / \Pr[I] \geq 1 + (5/8) / (1/4) = 7/2.$$

Finally, we obtain that $\Pr[F] \leq 3/8$ and $\Pr[C \mid \neg F] \geq 5/7$.

Now, let us run this algorithm 12ℓ times, on randomly self-reduced instances, and output the majority of the non-failure answers. On average, we get more than $u = 7.5\ell$ answers. The probability of error is upper-bounded by the probability to get more than $v = u/2$ incorrect answers among the u ones:

$$\begin{aligned} \Pr[\text{error}] &\leq \sum_{r=v+1}^u \binom{u}{r} \left(\frac{2}{7}\right)^r \left(\frac{5}{7}\right)^{u-r} = \left(\frac{5}{7}\right)^u \times \sum_{r=v+1}^u \binom{u}{r} \left(\frac{2}{5}\right)^r \\ &\leq \left(\frac{25}{49}\right)^v \left(\frac{2}{5}\right)^{v+1} \times \sum_{r=0}^v \binom{u}{r+v+1} \left(\frac{2}{5}\right)^r \\ &\leq \frac{2}{5} \times \left(\frac{10}{49}\right)^v \times \sum_{r=0}^v \binom{u}{r+v+1} \leq \frac{2}{5} \times \left(\frac{10}{49}\right)^v \times \frac{2^v}{2} \leq \frac{2}{5} \times \left(\frac{40}{49}\right)^v \end{aligned}$$

Finally, this probability is upper-bounded by $(1/2)^\ell$, since $v = 15\ell/4$. \square

Theorem 11 (The Strong Random Self-Reducible Problem Case). *Let us consider an attacker \mathcal{A} against the semantic security of Enc_f in a chosen-ciphertext scenario running within a time bound T . If we denote by ε the advantage of this attacker, one can design an algorithm \mathcal{B} that returns, for any given z , a partial preimage of z by f in an expected time bounded by $21\ell T/\delta$, with a negligible probability of error upper-bounded by $2^{-\ell}$, for any parameter ℓ , where*

$$\delta = \left(\varepsilon - \frac{q_H + q_D}{2^{k_1}} - \frac{q_D}{Y} \right) \approx \varepsilon.$$

Proof. It is an easy corollary of above lemma. Indeed, if one runs $7/8\delta$ times the general reduction (with randomly self-reduced instances), collecting all the output sets, the global set contains the correct solution with probability greater than $7/8$. \square

Another situation may exist where the verification of the rightness of the candidate is easy. In this case, the efficiency of the reduction is much better. Indeed, from the list of candidates, one has just to check if one of them is the solution.

Theorem 12 (The Easy Verifiable Case). *Let us consider an attacker \mathcal{A} against the semantic security of Enc_f in a chosen-ciphertext scenario. If we denote by ε the advantage of this attacker, one can design an algorithm \mathcal{B} which runs within almost the same time and outputs a partial preimage by f of any given z with probability greater than $\varepsilon - (q_H + q_D)/2^{k_1} - q_D/Y$.*

4 Applications

Let us apply this result to some encryption schemes to make provide semantic security, even against adaptively chosen-ciphertext attacks in the random oracle model, without any more assumption than the one-wayness of the original encryption scheme.

4.1 The El Gamal Encryption Scheme

If one applies our transformation to the famous El Gamal encryption scheme [15], which means to the Diffie-Hellman problem, one gets the scheme presented in Figure 3, together with the following security properties. As previously seen, the

$\mathcal{G} = \langle g \rangle$ of order q
$H : \{0, 1\}^k \rightarrow \mathbb{Z}_q$ and $G : \mathcal{G} \rightarrow \{0, 1\}^k$
Secret Key: $x \in \mathbb{Z}_q$
Public Key: $y = g^x$
Encryption
$r \in_R \mathcal{G}$ and $s \in_R \{0, 1\}^{k_1}$
$d = H(m \ s)$
$\text{Enc}(m, r \ s) = \begin{cases} a = g^d \\ b = y^d \cdot r \\ c = (m \ s) \oplus G(r) \end{cases}$
Decryption
$\text{Dec}(a, b, c) = \begin{cases} r = b/a^x & t = c \oplus G(r) \\ \text{if } a = g^{H(t)} & \text{then } m = [t]_{k_0} \end{cases}$

Fig. 3. The DH-based Encryption Scheme

partially trapdoor one-way injection is known as relying on the Computational Diffie-Hellman Problem:

$$\begin{aligned} f : \mathcal{G} \times \mathbb{Z}_q &\longrightarrow \mathcal{G} \times \mathcal{G} \\ (m, a) &\longmapsto (g^a, y^a \cdot m) \end{aligned} \quad \begin{aligned} g : \mathcal{G} \times \mathcal{G} &\longrightarrow \mathcal{G} \\ (x, z) &\longmapsto m = z/x^b \end{aligned}$$

Furthermore, it is well-known to be random self-reducible, even in our strong sense: for a given $(\alpha, \beta) = f(m, a)$, for random $u, v, w \in \mathbb{Z}_q$, $(\alpha^u g^v, \beta^u y^v g^w) = f(m^u g^w, au + v)$, with $(m^u g^w, au + v)$ uniformly distributed in $\mathcal{G} \times \mathbb{Z}_q$.

One has just to remark that, during the decryption phase, if $a = g^d \bmod p$, then $b = a^x r = y^d r \bmod p$ holds.

Theorem 13 (The DH-based Encryption Scheme). *Any algorithm \mathcal{A} able to break the semantic security of the DH-based Encryption Scheme under adaptively chosen-ciphertext attacks within time T can be used as a subroutine to an algorithm \mathcal{B} that breaks the Computational Diffie-Hellman problem in an expected time bounded by $30T\ell/\varepsilon$, with a negligible probability of error bounded by $2^{-\ell}$, for any parameter ℓ , where*

$$\varepsilon = \left(\text{Adv}_{\mathcal{A}} - \frac{q_D + q_H}{2^{k_1}} - \frac{q_D}{q} \right) \approx \text{Adv}_{\mathcal{A}}.$$

Advantages of the DH-based Encryption Scheme considered as an El Gamal variant. At PKC '98, Tsiounis and Yung [38] studied El Gamal based encryption schemes. They were the first to propose a variant secure against chosen-ciphertext attacks, in the random oracle model. However, it was also based on both the Decisional Diffie-Hellman problem and an unproven assumption about the unforgeability of Schnorr signatures [35]. Furthermore, for weaker schemes, it required more computations: three exponentiations instead of only two for both encryption and decryption in ours.

Later in the same year, Shoup and Gennaro [37] proposed a new variant provably secure against chosen-ciphertext attacks in the random oracle model, under the sole assumption of the Decisional Diffie-Hellman problem. Once again, efficiency was a serious backward: encryption required five exponentiations instead of two for ours, and decryption required seven exponentiations instead of two! However, it was the first convincing El Gamal variant.

Finally, one could consider the Fujisaki-Okamoto variant [16], with a similar efficiency in the random oracle model, or the Cramer-Shoup variant [12], twice slower but, for the first time, proven in the standard model. However, in both cases, security is relative to the Decisional Diffie-Hellman problem, a much stronger assumption than the Computational one.

Consequently, this El Gamal variant is the most efficient from our knowledge (only two exponentiations per encryption or decryption). Furthermore, it is semantically secure against adaptively chosen-ciphertext attacks under the sole assumption of the Computational Diffie-Hellman problem (and not the Decisional one), in the random oracle model.

4.2 The Okamoto-Uchiyama Encryption Scheme

Let us turn to the Okamoto-Uchiyama encryption scheme [24], which is one-way related to the factorization. Our transformation leads to the scheme presented in Figure 4, together with the following security properties. The partially trapdoor

p, q large prime integers of same length, and $n = p^2q$ $H : \{0, 1\}^k \rightarrow \mathbb{Z}_n$ and $G : \mathbb{Z}_n \rightarrow \{0, 1\}^k$ $g \in \mathbb{Z}_n^*$ such that the order of $g_p = g^{p-1} \pmod{p^2}$ is p $h = g^n \pmod{n}$
Encryption $r \in_R \mathbb{Z}_n, s \in_R \{0, 1\}^{k_1}$ $\mathcal{E}nc(m, r \ s) = \begin{cases} a = g^r h^{H(m \ s)} \\ b = (m \ s) \oplus G(r) \end{cases}$
Decryption $Dec(a, b, c) = \begin{cases} r = L(y_p)/L(g_p) \pmod{p} \\ m \ s = b \oplus G(r) \\ a \stackrel{?}{=} g^r h^{H(m \ s)} \end{cases}$ where $y_p = y^{p-1} \pmod{p^2}$ and $L(x) = (x - 1)/p$.

Fig. 4. The OU-based Encryption Scheme

one-way injection is known as relying on the factorization of the large integer $n = p^2q$:

$$f : \mathbb{Z}_p \times \mathbb{Z}_{(p-1)(q-1)} \longrightarrow \mathbb{Z}_n^* \quad g : \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_p \\ (x, r) \longmapsto g^x \times h^r \pmod{n} \quad y \longmapsto \frac{L(y_p)}{L(g_p)} \pmod{p}$$

This problem is well-known to be random self-reducible, however, one can furthermore use the “easy verifiable” property. Indeed, we can use the attacker to suggest candidates as partial-preimage of a known $y = g^x h^n \pmod{n}$, with a rather large x . With all the candidates a , one computes $\gcd(x - a, n)$ which should provide p for the right solution.

Theorem 14 (The OU-based Encryption Scheme). *Any algorithm \mathcal{A} able to break the semantic security of the OU-based Encryption Scheme under adaptively chosen-ciphertext attacks within time T can be used to factor n with probability greater than $\text{Adv}_{\mathcal{A}} - (q_H + q_D)/2^{k_1} - q_D/Y$, within the same time T .*

Advantages of the OU-based Encryption Scheme. The main advantage of this scheme is clear: the original one [24] is totally breakable under a (non-adaptive) chosen-ciphertext attack, which is a serious drawback. However its main interest was the factorization-based security. But just the one-wayness was related to the

factorization. Indeed, even semantic security against chosen-plaintext attacks requires the higher residues assumption.

The presented scheme does not increase so much the computational load, but considerably enhances the security: chosen-ciphertext security is related to factorization, by a *perfect reduction* (the underlying problem can be broken within the same time and identical probability as the security property).

5 Conclusion

In this paper, we have presented the most interesting generic transformation which provides chosen-ciphertext secure schemes from the weakest possible assumption: the existence of partially trapdoor one-way functions. Furthermore, the exact security provides very practical results in the most common cases, random self-reducible or easy verifiable problems. Indeed, in this latter case, the reduction is optimal: the underlying problem can be broken within the same time and with the same probability than the resulting encryption scheme.

Finally, applications to well-known problems lead to very useful schemes: the most efficient based on the Computational Diffie–Hellman problem and the first one as secure as factorization.

Furthermore, to improve efficiency, one can integrate symmetric encryption, with $G(r)$ as secret key, instead of using the one-time pad, as it has already been done in recent works [1, 25, 17, 30].

Acknowledgements

I would like to thank Dan Boneh for his help and many fruitful comments.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. IEEE P1363a Submission. September 1998. Available from <http://grouper.ieee.org/groups/1363/addendum.html>. 144
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998. 129, 132, 138
3. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993. 132, 135
4. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995. 129, 130, 132, 133, 137, 138
5. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996. 132

6. M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Crypto '99*, LNCS 1666, pages 519–536. Springer-Verlag, Berlin, 1999. [129](#), [132](#)
7. J. D. Cohen (Benaloh). Improving Privacy in Cryptographic Elections. Technical Report TR-454, Yale University, February 1986. [130](#), [133](#), [135](#)
8. J. D. Cohen (Benaloh). *Improving Privacy in Cryptographic Elections*. PhD thesis, Yale University, September 1987. Also available as technical report TR-561. [130](#), [133](#), [135](#)
9. D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In *Crypto '98*, LNCS 1462, pages 1–12. Springer-Verlag, Berlin, 1998. [130](#)
10. D. Coppersmith, S. Halevi, and C. S. Jutla. ISO 9796 and the New Forgery Strategy. Working Draft presented at the Rump Session of *Crypto '99*, 1999. [130](#)
11. S. Coron, D. Naccache, and J. P. Stern. On the Security of RSA Padding. In *Crypto '99*, LNCS 1666, pages 1–18. Springer-Verlag, Berlin, 1999. [130](#)
12. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Crypto '98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998. [130](#), [142](#)
13. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976. [129](#), [130](#), [133](#), [134](#)
14. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *Proc. of the 23rd STOC*. ACM Press, New York, 1991. [129](#), [130](#), [132](#)
15. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985. [130](#), [131](#), [132](#), [133](#), [134](#), [141](#)
16. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC '99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999. [130](#), [132](#), [142](#)
17. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999. [131](#), [132](#), [144](#)
18. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984. [130](#), [131](#)
19. SET Secure Electronic Transaction LLC. SET Secure Electronic Transaction Specification – Book 3: Formal Protocol Definition, may 1997.
Available from <http://www.setco.org/>. [130](#)
20. U. M. Maurer. Diffie Hellman Oracles. In *Crypto '96*, LNCS 1109, pages 268–282. Springer-Verlag, Berlin, 1996. [135](#)
21. D. Naccache and J. Stern. A New Cryptosystem based on Higher Residues. In *Proc. of the 5th CCS*, pages 59–66. ACM Press, New York, 1998. [130](#), [133](#), [135](#)
22. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437. ACM Press, New York, 1990. [130](#), [132](#)
23. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Crypto '98*, LNCS 1462, pages 354–369. Springer-Verlag, Berlin, 1998. [132](#)
24. T. Okamoto and S. Uchiyama. A New Public Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, Berlin, 1998. [130](#), [131](#), [133](#), [135](#), [139](#), [143](#)

25. T. Okamoto, S. Uchiyama, and E. Fujisaki. EPOC: Efficient Probabilistic Public-Key Encryption. Submission to IEEE P1363a. November 1998.
Available from <http://grouper.ieee.org/groups/1363/addendum.html>. [132](#), [144](#)
26. P. Paillier. A Trapdoor Permutation Equivalent to Factoring. In *PKC '99*, LNCS 1560, pages 219–222. Springer-Verlag, Berlin, 1999. [130](#)
27. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, Berlin, 1999. [130](#), [131](#), [133](#), [135](#), [139](#)
28. P. Paillier and D. Pointcheval. Efficient Public-Key Cryptosystems Provably Secure against Active Adversaries. In *Asiacrypt '99*, LNCS. Springer-Verlag, Berlin, 1999. [130](#), [132](#)
29. D. Pointcheval. New Public Key Cryptosystems based on the Dependent-RSA Problems. In *Eurocrypt '99*, LNCS 1592, pages 239–254. Springer-Verlag, Berlin, 1999. [130](#), [132](#)
30. D. Pointcheval. HD-RSA: Hybrid Dependent RSA - a New Public Key Encryption Scheme. Submission to IEEE P1363a. October 1999.
Available from <http://grouper.ieee.org/groups/1363/addendum.html>. [130](#), [144](#)
31. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 1999.
Available from <http://www.di.ens.fr/~pointche>. [132](#)
32. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992. [129](#), [130](#), [132](#)
33. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. [130](#), [131](#), [132](#)
34. RSA Data Security, Inc. Public Key Cryptography Standards – PKCS.
Available from <http://www.rsa.com/rsalabs/pubs/PKCS/>. [130](#)
35. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991. [130](#), [142](#)
36. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266. Springer-Verlag, Berlin, 1997. [139](#), [140](#)
37. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt '98*, LNCS 1403, pages 1–16. Springer-Verlag, Berlin, 1998. [130](#), [132](#), [142](#)
38. Y. Tsounis and M. Yung. On the Security of El Gamal based Encryption. In *PKC '98*, LNCS. Springer-Verlag, Berlin, 1998. [130](#), [142](#)

Short Proofs of Knowledge for Factoring

Guillaume Poupard and Jacques Stern

École Normale Supérieure, Département d’Informatique,
45 rue d’Ulm, F-75230 Paris Cedex 05, France
`{Guillaume.Poupard,Jacques.Stern}@ens.fr`

Abstract. The aim of this paper is to design a proof of knowledge for the factorization of an integer n . We propose a statistical zero-knowledge protocol similar to proofs of knowledge of discrete logarithm *a la* Schnorr. The efficiency improvement in comparison with the previously known schemes can be compared with the difference between the Fiat-Shamir scheme and the Schnorr one. Furthermore, the proof can be made non-interactive. From a practical point of view, the improvement is dramatic: the size of such a non-interactive proof is comparable to the size of the integer n and the computational resources needed can be kept low; three modular exponentiations both for the prover and the verifier are enough to reach a high level of security.

1 Introduction

Zero-knowledge (ZK) proofs have first been proposed in 1985 by Goldwasser, Micali and Rackoff [14]. Those proofs are interactive protocols between a prover who wants to convince a verifier that an object belongs to a language (proof of membership) or that he knows a secret information (proof of knowledge), without revealing anything about his secret knowledge. Such proofs have practical applications since they allow to solve many cryptographic problems such as ZK identification [9], digital signature [25] or robust distributed cryptography [26]. Many ZK proof systems have been published so far that are related to the presumably intractable problems on which public key cryptography is based, such as the computation of discrete logarithms [25], of square roots [9] and of e^{th} roots [15] modulo a composite integer.

In this paper, we consider the most popular such problem: the factorization of integers, i.e. how to prove to a verifier that one’s knows some prime numbers whose product is a public number without giving any information about this decomposition. Proofs of knowledge for the factorization of an integer have been known for a long time. But, even if they are claimed efficient according to complexity theoretical arguments, none of them can be considered practical for many applications because of their significant communication complexity: the proof is much longer than the object it deals with.

Previously known proofs of knowledge for the factorization of an integer n are closely related to the property that n has less than a prescribed number of

prime factors. Van de Graaf and Peralta [29] and Galil et al. [11] first provided efficient proofs that a given integer is of the form $p^r q^s$. Then Boyar et al. [4] proposed a protocol to prove that an integer is square-free. The combination of those two protocols typically allows to prove that an integer is an RSA modulus. Those results have recently been enhanced with protocols which prove that the factors are *quasi-safe* primes [12], have about the same size [16] or are exactly safe primes [5].

All the above protocols are based on the basic observation that, modulo a given integer n , a random number has a square root with probability $2^{-\eta}$, where η is the number of different prime factors of n , and that such a square root can be efficiently computed only when the factorization of n is known. A verifier can ask for square roots of randomly chosen numbers and count the proportion of correct answers of the prover. This allows to prove that a number has less than η different prime factors but the basic round has to be repeated many times in order to reach a sufficient level of security and confidence.

A derived protocol to prove the knowledge of the factorization of n has been proposed by Tompa and Woll [28]. It is based on ZK proofs for quadratic residuosity that come from [14] and which can be interpreted as proofs of membership of an integer x to the set of the quadratic residues modulo an integer n or as proofs of knowledge of a square root of x modulo n . According to Tompa and Woll, the verifier first randomly chooses an element r in \mathbb{Z}_n and sends $x = r^2 \bmod n$ to the prover. Then he proves that x is a quadratic residue modulo n . Once the prover is convinced that the verifier knows a square root of x , himself computes a square root s of x and proves that he also knows a square root of x .

The completeness of such a protocol is based on the existence of a polynomial time algorithm for computing square roots modulo n when the factorization of n is known. Furthermore, this scheme is zero-knowledge since the view of any prover can be easily simulated using the simulator of the proof for quadratic residuosity proposed in [14]. This explains why the verifier needs to prove that he knows a square root of x ; otherwise, he would learn whether x is a quadratic residue or not and the protocol would not be zero-knowledge. Finally, the scheme has error probability smaller than $7/8$ but greater than $3/4$. Consequently, it has to be repeated many times in order to reach a high level of security. For realistic parameters, this implies Giga-bytes of communication. From a theoretical point of view, let k be a security parameter such that the cheating probability is smaller than $1/2^k$. We can prove that the complexity for both computation and communication is $\theta(k \times |n|^2)$ where $|n|$ denotes the number of digits in the binary expansion of n . Even if some optimizations can be added, it does not seem possible to go under $\theta(k \times |n|)$.

Notice that Boudot and Traoré [3] have recently proposed a scheme that does not need to prove that n has less than a prescribed number of prime factors. The idea, somehow comparable with what we do in this paper, is to prove the knowledge of a common discrete logarithm d of g_1 and g_2 , two randomly chosen integers, in basis g_1^e and g_2^e , where d is such that $e \times d = 1 \bmod \lambda(n)$.

1.1 Our Results

We propose an interactive proof of knowledge for the factorization of any public integer n whose prime factors cannot be found by simple trial division. It is statistical zero-knowledge. The protocol is a proof of knowledge of a small discrete logarithm of $z^n \bmod n$ for a few randomly chosen elements z modulo n . Consequently, it is related to variants of the Schnorr proof of knowledge for discrete logarithms.

Our scheme is very efficient. When suitably optimized, its communication complexity is only $O(k + |n|)$ bits. In this setting, the size of our proof is similar to the size of the integer n . The improvement in comparison with the previously known schemes can therefore be compared with the difference of efficiency between the Fiat-Shamir scheme and the Schnorr one. Furthermore, the computational complexity is proved to be $O((|n| + k) \times k)$ multiplications modulo n both for the prover and the verifier but we provide strong heuristic evidence to show that $O((|n| + k) \times k / \log k)$ is enough. This might appear a small improvement but it has drastic consequences in practical terms: only three modular exponentiations both for the prover and the verifier are needed to obtain a very high level of security.

In section 2 we state results of independent interest on the probability to generate the multiplicative group $\mathbb{Z}_{p^e}^*$ with few elements (complete proofs appear in the appendix A). Those results are used in the security analysis of the interactive proof of knowledge we describe in section 3. We also propose a communication efficient variant (section 3.3) and a non-interactive version (section 3.4). In section 4 we give heuristic arguments to improve the computational efficiency of the scheme and, in section 5, we show the practical efficiency of those proofs in actual applications. Finally, in section 6, we propose a variant suggested by Adi Shamir in order to make the proof work even when n has a small prime factors.

1.2 Notations and Definitions

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 $\lambda(n) = \text{lcm}_{i=1 \dots \eta} (q_i^{f_i-1}(q_i - 1))$.

If $(g_i)_{i \in [1, K]}$ is a K -tuple of $(\mathbb{Z}_n^*)^K$, we use $\langle (g_i)_{i \in [1, K]} \rangle$ to denote the subgroup of \mathbb{Z}_n^* that is generated by the g_i s, i.e.

$$\left\langle (g_i)_{i \in [1, K]} \right\rangle = \left\{ x \in \mathbb{Z}_n^* \mid \exists(\lambda_1, \dots, \lambda_K) \quad x = \prod_{i=1}^K g_i^{\lambda_i} \bmod n \right\}$$

In the following, p_i is the i^{th} prime number ($p_1 = 2, p_2 = 3, \dots$). For any finite set S , $\text{Card}(S)$ denotes the number of elements of S . We finally denote by $\zeta(K)$ the Riemann Zeta function defined by $\zeta(K) = \sum_{d=1}^{+\infty} \frac{1}{d^K}$ for any integer $K \geq 2$.

2 On the Generation of $\mathbb{Z}_{p^e}^*$

We state known facts about the generation of the cyclic multiplicative group $\mathbb{Z}_{p^e}^*$ where p is an odd prime number and $e \geq 1$. Using generators it is possible to precisely estimate the probability to generate the full group $\mathbb{Z}_{p^e}^*$ by a single randomly chosen element.

Theorem 1 *For any prime number $p \geq 7$, for any $e \geq 1$,*

$$\Pr_{g \in \mathbb{Z}_{p^e}^*} \{ \langle g \rangle = \mathbb{Z}_{p^e}^* \} = \frac{\varphi(\varphi(p^e))}{\varphi(p^e)} > \frac{1}{7 \ln \ln p}$$

Next, we generalize this result when K elements are randomly chosen instead of one. We obtain the following lower bound, independent of p and e :

Theorem 2 *For any odd prime number p , for any $e \geq 1$, for any $K \geq 2$,*

$$\Pr_{\{g_i\}_{i \in [1, K]} \in (\mathbb{Z}_{p^e}^*)^K} \left\{ \left\langle (g_i)_{i \in [1, K]} \right\rangle = \mathbb{Z}_{p^e}^* \right\} > \frac{1}{\zeta(K)} > 1 - \frac{K+1}{K-1} \times \frac{1}{2^K}$$

Finally, we obtain a lower bound for the probability that K elements generate a *large* subgroup of $\mathbb{Z}_{p^e}^*$, i.e. a subgroup of size greater than $\text{Card}(\mathbb{Z}_{p^e}^*) / C$ for a fixed parameter C :

Theorem 3 *For any odd prime number p , for any $e \geq 1$, $C \geq 1$ and $K \geq 2$,*

$$\text{with } \mathcal{P} = \Pr_{\{g_i\}_{i \in [1, K]} \in (\mathbb{Z}_{p^e}^*)^K} \left\{ \text{Card} \left\langle (g_i)_{i \in [1, K]} \right\rangle \geq \frac{\text{Card}(\mathbb{Z}_{p^e}^*)}{C} \right\}$$

$$\mathcal{P} > \frac{1}{\zeta(K)} \times \sum_{d=1}^C \frac{1}{d^K} > 1 - \frac{1}{(K-1)C^{K-1}\zeta(K)}$$

All the proofs appear in appendix A.

3 Proofs of Knowledge for Factoring

3.1 Description

Let k be a security parameter. Let n be an integer whose number of digits in its binary expansion is denoted $|n|$. Let A, B, ℓ and K be integers which depend *a priori* on k and $|n|$. Let z_1, \dots, z_K be K elements randomly chosen in \mathbb{Z}_n^* . We describe an interactive proof of knowledge for the factorization of n .

A round of proof (see figure 1) consists for the prover in randomly choosing an integer r in $[0, A[$ and computing, for $i = 1..K$, the *commitments* $x_i = z_i^r \bmod n$. Then he sends the x_i s to the verifier who answers a *challenge* e randomly chosen in $[0, B[$. The prover computes $y = r + (n - \varphi(n)) \times e$ (in \mathbb{Z}) and sends it to the verifier who checks $0 \leq y < A$ and, for $i = 1..K$, $z_i^{y-n \times e} = x_i \bmod n$. A complete proof consists in repeating ℓ times the elementary round.

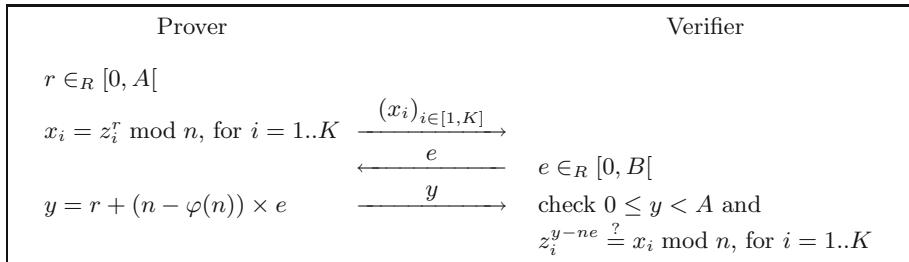


Fig. 1. Interactive proof of knowledge for factoring (elementary round)

This scheme is a variant of the Schnorr proof of knowledge of discrete logarithms. It consists in proving the knowledge of small discrete log, namely $n - \varphi(n)$, of $z_i^n \bmod n$, for K randomly chosen integers z_i in \mathbb{Z}_n^* . In the following section, we prove that it is a statistical ZK proof of knowledge of the factorization of n .

3.2 Security Proofs

In order to prove the security of the protocol, we follow the approach of Feige, Fiat and Shamir [8], first proving completeness, then soundness and, finally, the zero-knowledge property. In order to simplify the notations, we do not write the dependencies on k but when we say that an expression f is negligible, this means that f depends on k and that, for any constant c and for large enough k , $f(k) < 1/k^c$. Our computing model is the probabilistic polynomial time Turing machine (PPTM), whose running time is a polynomial in k and $|n|$.

Theorem 4 (Completeness) *The execution of the protocol between a prover who knows the factorization of n and a verifier is successful with overwhelming probability if $(n - \varphi(n))\ell B/A$ is negligible.*

Proof: At the end of each round, the verifier obtains $x_i = z_i^r \bmod n$ and $y = r + (n - \varphi(n)) \times e$ which can be easily computed by the prover if he knows the factorization of n . From Euler's theorem, we know that $z_i^{\varphi(n)} = 1 \bmod n$ so $z_i^y = z_i^{r+(n-\varphi(n))e} = x_i \times z_i^{ne} \bmod n$. Consequently, $z_i^{y-n \times e} = x_i \bmod n$.

If the prover follows the protocol, the proof fails only if $y \geq A$ at some round of the proof. The probability of failure of such an event taken over all possible choices of r is smaller than $(n - \varphi(n))B/A$. Consequently the execution of the protocol is successful with probability $\geq \left(1 - \frac{(n-\varphi(n))B}{A}\right)^\ell \geq 1 - \frac{(n-\varphi(n))\ell B}{A}$. Thus, if $(n - \varphi(n))\ell B/A$ is negligible, the probability of success is overwhelming. \square

The proof of soundness consists in proving that, if the verifier accepts the proof, then, with overwhelming probability, the prover must know the factorization of n . Intuitively, after the commitment of the x_i s, if the prover is accepted with probability $> 1/B$, he must be able to answer two different questions e and e' with y and y' smaller than A such that $z_i^{y-ne} = x_i = z_i^{y'-ne'} \bmod n$ for $i = 1..K$. Let $\lambda_0 = |(y - y') - n(e - e')|$; this integer is such that, for $i = 1..K$, $z_i^{\lambda_0} = 1 \bmod n$. The following lemma formally states those ideas, where ε is implicitly assumed to depend on k and $|n|$:

Lemma 1 *Assume that some PPTM adversary \tilde{P} is accepted with probability $\varepsilon' = 1/B^\ell + \varepsilon$, $\varepsilon > 0$ and that $A < n$. Then there exists an algorithm which, with probability $> \varepsilon^2/(6\varepsilon'^2)$, outputs $\lambda_0 \in]0, A + nB]$ such that, for $i = 1..K$, $z_i^{\lambda_0} = 1 \bmod n$. The expected running time is $< 2/\varepsilon \times \tau$, where τ is the average running time of an execution of the proof.*

Proof: Assume that some PPTM adversary $\tilde{P}(\omega)$, running on random tape ω , is accepted with probability $\varepsilon' = 1/B^\ell + \varepsilon$. We write $Succ(\omega, (e_1, \dots, e_\ell)) \in \{\text{true}, \text{false}\}$ the result (successful or not) of the identification of $\tilde{P}(\omega)$ when the challenges e_1, \dots, e_ℓ are used.

We consider the following algorithm (largely inspired from [25]):
Step 1. Pick a random tape ω and a tuple e of ℓ integers e_1, \dots, e_ℓ in $\{0, \dots, B-1\}$ until $Succ(\omega, e)$. Let u be the number of probes.

Step 2. Probe up to u random ℓ -tuples e' different from e until $Succ(\omega, e')$. If after the u probes a successful e' is not found, the algorithm fails.

Step 3. Let j be one of the indices such that $e_j \neq e'_j$; we note y_j and y'_j the related correct answers of \tilde{P} . The algorithm outputs $\lambda_0 = |(y_j - y'_j) - n(e_j - e'_j)|$.

If this algorithm does not fail, the prover is able to correctly answer two challenges e_j and e'_j for the same commitment x_j with the answers y_j and y'_j . This means that $z_i^{y_j - n \times e_j} = x_j = z_i^{y'_j - n \times e'_j} \bmod n$ for all $i = 1..K$ so the integer λ_0 is such that $z_i^{\lambda_0} = 1 \bmod n$. Furthermore, λ_0 is smaller than $A + nB$ because $\lambda_0 = |(y_j - y'_j) - n(e_j - e'_j)|$ for integers y_j and y'_j smaller than A and integers e_j and e'_j smaller than B . Finally, since $A < n$, $\lambda_0 = 0$ would imply $e_j = e'_j$ so $\lambda_0 > 0$.

We now analyze the complexity of the algorithm. By assumption, the probability of success of \tilde{P} is ε' so the first step finds ω and e with the same probability.

The expected number E of repetitions is $1/\varepsilon'$ and the number u of probes is equal to N with probability $\varepsilon' \times (1 - \varepsilon')^{N-1}$.

Let Ω be the set of random tapes ω such that $\Pr_e \{Succ(\omega, e)\} \geq \varepsilon' - \varepsilon/2 = 1/B^\ell + \varepsilon/2$. The probability for the random tape ω found in step 1 to be in Ω conditioned by the knowledge that $Succ(\omega, e) = \text{true}$ can be lower bounded:

$$\begin{aligned} \Pr_{\omega, e} \{\omega \in \Omega | Succ(\omega, e)\} &= 1 - \Pr_{\omega, e} \{\omega \notin \Omega | Succ(\omega, e)\} \\ &= 1 - \Pr_{\omega, e} \{Succ(\omega, e) | \omega \notin \Omega\} \times \frac{\Pr_{\omega, e} \{\omega \notin \Omega\}}{\Pr_{\omega, e} \{Succ(\omega, e)\}} \\ &\geq 1 - \left(\frac{1}{B^\ell} + \frac{\varepsilon}{2} \right) \times 1/\varepsilon' = \frac{\varepsilon}{2 \times \varepsilon'} \end{aligned}$$

With probability $> \varepsilon/(2\varepsilon')$, the random tape ω is in Ω and in this case, by definition of the set Ω , the probability for a tuple of challenges $e' \neq e$ to lead to success is $\geq \varepsilon/2$. The probability to obtain such a tuple e' after less than N probes is $\geq 1 - (1 - \varepsilon/2)^N$.

Consequently, the probability to obtain a random tape ω in Ω and to find e' is greater than

$$\frac{\varepsilon}{2\varepsilon'} \times \sum_{N=1}^{+\infty} (1 - \varepsilon')^{N-1} \times \varepsilon' \times \left[1 - \left(1 - \frac{\varepsilon}{2}\right)^N \right] = \frac{\varepsilon^2}{4\varepsilon'(\varepsilon' + \varepsilon/2 - \varepsilon \times \varepsilon'/2)} > \frac{\varepsilon^2}{6\varepsilon'^2}$$

In conclusion, the algorithm finds λ_0 with probability $> \varepsilon^2/(6\varepsilon'^2)$ and the total expected number of executions of the proof between \tilde{P} and a verifier is smaller than $2/\varepsilon'$. \square

Theorem 5 (Soundness) *Assume that some PPTM adversary \tilde{P} is accepted with non-negligible probability. If $\ell \times \log B = \theta(k)$, $K = \theta(k + \log(|n|))$, $\log(A)$ is a polynomial in k and $|n|$ and $A < n$, there exists a PPTM which factors n with overwhelming probability.*

Proof: Let $\pi(k)$ is the probability of success of \tilde{P} . If $\pi(k)$ is non-negligible, there exists an integer d such that $\pi(k) \geq 1/k^d$ for infinitely many values k .

Let $n = \prod_{j=1}^{\eta} q_j^{e_j}$ be the prime factorization of n . Notice that η is the number of different prime factors of n . Let us consider the K randomly chosen elements z_i ; from theorem 2, we know that, modulo $q_j^{e_j}$, they generate $\mathbb{Z}_{q_j^{e_j}}^*$ with probability greater than $1 - (K+1)/(K-1) \times 1/2^K$. Consequently, the z_i s generate multiplicative groups modulo $q_j^{e_j}$ for $j = 1..n$ with probability greater than $1 - \eta \times (K+1)/(K-1) \times 1/2^K$.

The probability for \tilde{P} to be accepted while the z_i s generate all groups $\mathbb{Z}_{q_j^{e_j}}^*$ is larger than $\pi(k) - \eta \times (K+1)/(K-1) \times 1/2^K$. The number η of different prime factors of n is less than $\log_2(n) = |n|$ so, if $K = \theta(k + \log(|n|))$, for infinitely many values k , $\eta \times (K+1)/(K-1) \times 1/2^K \leq 1/3k^d$.

Furthermore, for k large enough, $1/B^\ell < 1/3k^d$ if $\ell \times \log B = \theta(k)$. So, taking $\varepsilon = \pi(k)/3$ in lemma 1 we conclude that it is possible to obtain $\lambda_0 \in]0, A+nB]$ in polynomial time $O(1/\varepsilon) = O(k^d)$.

Then, for $i = 1..K$, $z_i^{\lambda_0} = 1 \bmod n$ so, for any $j = 1..\eta$, $z_i^{\lambda_0} = 1 \bmod q_j^{e_j}$. Let z be any element of \mathbb{Z}_n^* ; since the z_i s generate $\mathbb{Z}_{q_j^{e_j}}^*$ for a fixed j , z can be written as a product $\prod_{i=1}^K z_i^{\alpha_{i,j}}$ modulo $q_j^{e_j}$ and consequently $z^{\lambda_0} = 1 \bmod q_j^{e_j}$. Using the Chinese remainder theorem, we obtain that $z^{\lambda_0} = 1 \bmod n$ for any z in \mathbb{Z}_n^* . This means that λ_0 is a non-zero multiple of the Carmichael lambda function of n . It is well known that knowledge of a multiple of $\lambda(n)$ allows to factor n in time $O(\eta \times \log \lambda_0)$ modular multiplications using the Miller's factoring algorithm [18] which we recall in appendix B.

Finally, we obtain the factorization of n in time $O(|n| \times \log(A+nB))$ modular multiplications modulo n . \square

Theorem 6 (Zero-Knowledge) *The protocol is statistically zero-knowledge if $(n - \varphi(n))\ell B/A$ is negligible and $\ell \times B$ is a polynomial in k .*

Proof: We first remind that the zero-knowledge property is verified if the *view* of any verifier considered as a random variable is perfectly approximable by the output of a PPTM which does not know the factorization of n . A protocol is only *statistically zero-knowledge* if the view and the output of the PPTM are only statistically indistinguishable. We refer the reader to [14] for more details.

We describe the polynomial time simulation of the communication between a prover P and a dishonest verifier \tilde{V} . We assume that, in order to try to obtain information about the factorization of n , \tilde{V} does not randomly choose the challenges. If we focus on the j^{th} round of identification, \tilde{V} has already obtained data, noted $Data_j$, from previous interactions with P . Then the prover sends the commitments $X_j = (x_1, \dots, x_K)$ and \tilde{V} chooses, possibly using $Data_j$ and X_j , the challenge $e_j(Data_j, X_j)$.

Here is a simulation of the j^{th} round of identification: choose random values $e_j' \in [0, B[$ and $y_j' \in [0, A[$, compute, for $i = 1..K$, $x_i' = z_i^{y_j' - ne_j'} \bmod n$. If $e_j(Data_j, (x_1', \dots, x_K')) \neq e_j'$ then try again with another pair (e_j', y_j') , else return $((x_1', \dots, x_K'), e_j', y_j')$.

We observe that a good triplet $((x_1', \dots, x_K'), e_j', y_j')$ is obtained with probability $1/B$. Consequently, the expected time complexity of the all simulation is $O(\ell B)$.

Furthermore, it can be formally proved that such a simulation is statistically indistinguishable from the transcript of a real proof if $(n - \varphi(n))\ell B/A$ is negligible. Therefore, a verifier with infinite computation power cannot learn significant information after a polynomial number of authentications. \square

Note. This theorem shows that if we choose $\ell = 1$ and B exponential in the security parameter k , we cannot prove the zero-knowledge property. Notice that there is exactly the same problem with the Schnorr scheme.

Choice of the parameters and Complexity of the scheme. The choice of the parameters ℓ and B must be such that $\ell \times \log B = \theta(k)$ in order to make the protocol sound. The choice of A is a bit more difficult; A must be much larger than $(n - \varphi(n))\ell B$ to guarantee the completeness and the zero-knowledge property but A must also be smaller than n to guarantee the soundness. Consequently, n must verify $(n - \varphi(n)) \times 2^k \ll n$. The proof we propose cannot be used with any integer n but we can notice that the previous equation is verified by all the integers which does not have small prime factors. More precisely, if $n = \prod_{i=1}^{\eta} q_i^{e_i}$, we can prove that $\frac{1}{q_1} < \frac{n - \varphi(n)}{n} = 1 - \prod_{i=1}^{\eta} \left(1 - \frac{1}{q_i}\right) < \sum_{i=1}^{\eta} \frac{1}{q_i}$. Consequently, if $(n - \varphi(n)) \times 2^k \ll n$, all the prime factors of n must be $\gg 2^k$.

If all the prime factors of n are greater than a bound $F(k)$, we know that $(n - \varphi(n))/n < \eta/F(k)$ so we require $F(k) \gg \eta \times 2^k$. Anyway, in practical applications, such a proof is used to prove the knowledge of integers like RSA modulus with large prime factors; if n has small prime factors, the proof is not zero-knowledge but n can not be considered as a good modulus! Informally, this means that the proof is correct and zero-knowledge if the factorization of n is intractable. Notice that a prover cannot try to cheat choosing an integer n with small factors since the soundness is guaranteed by $A < n$.

The execution of the protocol requires the transmission of exactly $\ell \times (K \times |n| + |B| + |A|)$ bits. If we assume that $\ell \times \log B = \theta(k)$, $K = \theta(k + \log |n|)$ and $|A| = \theta(|n|)$ (with $A < n$), we obtain a communication complexity equal to $\theta(\ell \times (k + \log |n|) \times |n|)$. From the computational point of view, both the prover and the verifier need to compute K exponentiations modulo n with $|A|$ -bits exponents.

3.3 Optimized Version

In the interactive proof of knowledge we have described in section 3.1, we can observe that the largest part of the communication concerns the commitments x_i . Using an idea of Fiat and Shamir whose security has been formalized by Girault and Stern in [13], we can replace those commitments by the hash value $H(x_1, \dots, x_K)$ where H is an appropriate collision-free hash function. We obtain a new scheme (see figure 2), much more efficient in term of communication than the initial one. An important consequence is that the communication complexity of the modified protocol is independent of the parameter K and is the same than for the Schnorr scheme, i.e. $O(k + |n|)$.

We can finally observe that the commitments can be precomputed in order to reduce the on-line computation to a very simple non-modular arithmetic operation (like in [23]).

In practical applications, an important point is the choice of the z_i s. They need to be randomly chosen in order to make the proof sound. A first solution consists in using a mutually trusted source of random bits by the prover and the verifier. Such a strategy is used in non-interactive zero-knowledge proofs [2]. In practice, z_i can be pseudo-randomly generated from a seed of the form $h(n, i)$ where h is a hash function such as SHA-1 [19].

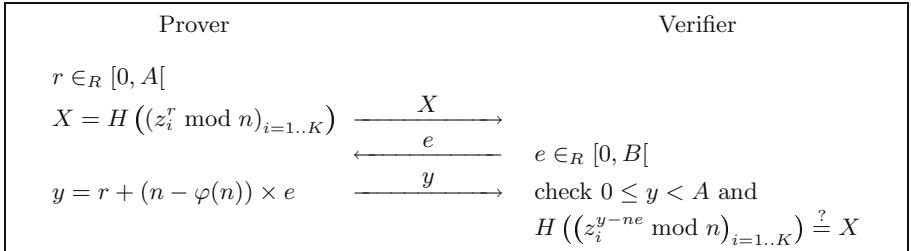


Fig. 2. Optimized interactive proof of knowledge for factoring

3.4 Non-interactive Proof of Knowledge for Factoring

The interactive proof can be made non-interactive using the Fiat-Shamir heuristic [8,9]. The verifier's challenge e is replaced with the hash value of the commitment $X = H(x_1, \dots, x_k)$ and of the public data using a collision-resistant hash function H' . The size of such a proof, $< k + |n|$ bits, is very small:

A non-interactive proof of knowledge of the factorization of n is
a pair (e, y) with $0 \leq e < B$, $0 \leq y < A$ and
 $e = H'(n, z_1, \dots, z_K, H(z_1^{y-ne} \bmod n, \dots, z_K^{y-ne} \bmod n))$

It is widely believed that such a transformation guarantees an accurate level of security as soon as H is random enough. Furthermore, the security of this approach can be formalized using the random oracle model [1,20,21] even if such analysis cannot be considered as an absolute proof of security [6].

The important difference between the interactive and the non-interactive setting is that in the second case we do not have to care about dishonest verifiers since the hash function H' is assumed to produce random challenges. It is easy to modify the proof of theorem 6 in order to demonstrate that if $\ell = 1$ and $B = 2^k$, the protocol is *honest-verifier statistically zero-knowledge* :

Theorem 7 *The protocol is honest-verifier statistically zero-knowledge if $(n - \varphi(n))\ell B/A$ is negligible.*

Then the so-called forking lemma technique described by Pointcheval and Stern [20,21] can be applied in order to prove the security of the non-interactive version of the scheme, in the random oracle model, even when $\ell = 1$.

4 Heuristic Efficiency Improvements

We have seen in the previous section that the proof of knowledge we propose has a communication complexity independent of the number K of integers z_i . Furthermore, for security reasons (theorem 5), K must be approximately equal to the security parameter k . In order to reduce the computational complexity

of the protocol, we now show how to reduce K to a very small value such as 3 in practice. The underlying idea is to increase the work load of the extractor of theorem 5 by a factor \sqrt{C} in order to reduce K by a factor $\log C$. Unfortunately, the algorithm we propose is based on well known techniques such as the Pollard's rho method whose complexity can only be analyzed using heuristic arguments.

Theorem 8 *Assume that some adversary is accepted with non-negligible probability ε . If $\ell \times \log B = \theta(k)$ and $K = \theta\left(\frac{k + \log(|n|)}{\log C}\right)$, there exists an algorithm which, heuristically, factors n in time $O(1/\varepsilon + \sqrt{C} \times |n| \times \log(A + nB))$ with non-negligible probability.*

If C is a polynomial in the security parameter k , this proves that we can choose $K = \theta((k + \log |n|)/\log k)$.

Let us first remind two algorithms.

Floyd's cycle-finding algorithm

Let f be a random function from a set S to S . Let x_0 be a random element of S and consider the elements x_i recursively defined by $x_{i+1} = f(x_i)$. Such a sequence consists of a *tail* of expected length $\sqrt{\pi \text{Card}(S)}/8$ followed by a *cycle* of the same expected length (see for example [10]). This immediately leads to an algorithm able to find a collision $x_i = x_j$ in time $O(\sqrt{\text{Card}(S)})$ and with memory $O(\sqrt{\text{Card}(S)})$ (see [17] for more details).

The previous algorithm can be improved in order to use just a constant amount of memory. Floyd's cycle-finding algorithm consists in starting with the pair (x_1, x_2) and iteratively computing (x_i, x_{2i}) from (x_{i-1}, x_{2i-2}) until $x_m = x_{2m}$. It can be proved that the expected running time of this algorithm is also $O(\sqrt{\text{Card}(S)})$ while the memory needed is constant since no values have to be stored in memory.

This algorithm can still be improved in order to find indexes i and j such that $x_i = x_j$ but $x_{i-1} \neq x_{j-1}$. The idea consists in finding in a first step an index m such that $x_m = x_{2m}$ with Floyd's algorithm. Then, we iteratively test if $x_i = x_{i+m}$ for increasing values of i . The time complexity is always $O(\sqrt{\text{Card}(S)})$ and the memory needed is still constant.

Pollard's rho factoring algorithm

Floyd's algorithm can be used to factor integers. The Pollard's rho algorithm [22] consists in choosing $S = \mathbb{Z}_n$ and $f(x) = x^2 + 1 \bmod n$. We do not search collisions $x_i = x_j \bmod n$ but only indexes i and j such that $\gcd(x_i - x_j, n) > 1$, i.e. a collision modulo a prime factor of n . Since this gcd is equal to n with negligible probability, we obtain a non trivial factor of n . A recursive use of the algorithm allows to completely factor n .

If we assume that $f(x) = x^2 + 1 \bmod p$ behaves like a random function, the computational complexity required to find a factor p of n is $O(\sqrt{p})$ modular multiplication. As a consequence, this algorithm allows to find small factors much more efficiently than with trial division.

Proof of theorem 8

We now describe the algorithm announced in theorem 8. The procedure we explain allows to break the integer n in two factors. A complete factorization of n is obtained by using it recursively.

We have proved in section 2 that the z_i s generate large subgroups modulo each prime power factor of n with overwhelming probability, even for small values of K , but we do not have any similar result modulo n . We note G the multiplicative group of the integers $z^{\lambda_0} \bmod n$ for $z \in \mathbb{Z}_n^*$. We know that $\text{Card}\{z \text{ s.t. } z^{\lambda_0} = 1 \bmod n\} \times \text{Card } G = \varphi(n)$. Consequently, $\Pr_{z \in \mathbb{Z}_n^*} \{z^{\lambda_0} = 1 \bmod n\} = 1/\text{Card } G$.

Two cases may occur:

- if $\text{Card } G$ is a *small* set, a multiple of $\lambda(n)$ can be computed from λ_0 . This can be done using a Floyd's algorithm to compute the order of $x^{\lambda_0} \bmod n$ for a randomly chosen elements x of \mathbb{Z}_n^* . Such an algorithm succeeds in expected time $O(\sqrt{\lambda(n)}/\gcd(\lambda_0, \lambda(n)))$. Then we can use the Miller's factoring algorithm to factor n with a multiple of $\lambda(n)$.
- otherwise, λ_0 does not have enough common factors with $\lambda(n)$ to make the Carmichael's lambda function of n easy to compute so we use the following algorithm to overcome the problem.

First, it is easy to test if the modulus n has more than η prime factors using the elliptic curve factoring algorithm or just the Pollard's rho factoring algorithm as soon as the size of the factors are *small enough*. For example, for 1024 bits modulus, $\eta = 16$ is reasonable.

Let $n = \prod_{j=1}^{\eta} q_j^{e_j}$ be the prime factorization of the modulus n . From theorem 3, we know that, modulo $q_j^{e_j}$, the K elements z_1, \dots, z_K generate a subgroup of $\mathbb{Z}_{q_j^{e_j}}^*$ which size is greater than $\varphi(q_j^{e_j})/C$ with probability greater than $1 - ((K-1)C^{K-1}\zeta(K))^{-1}$. Consequently, with probability greater than $1 - \eta \times ((K-1)C^{K-1}\zeta(K))^{-1}$, the z_i s generate large subgroups modulo each $q_j^{e_j}$. For example, with $\eta = 16$, $K = 3$ and $C = 2^{42}$, this probability is larger than $1 - 1/2^{80}$.

We now use a variant of Pollard's rho algorithm to factor n . Let z be a randomly chosen element of \mathbb{Z}_n^* . We define a sequence of elements modulo n by $w_0 = z^{\lambda_0} \bmod n$ and $w_{i+1} = (w_i^2 + 1)^{\lambda_0} \bmod n$. Then we look for indexes i and j such that $\gcd(w_i \pm w_j, n) \neq 1$ and $\gcd(w_{i-1} \pm w_{j-1}, n) = 1$ using the Floyd's cycle-finding algorithm.

Heuristically, since the cycles modulo each factor $q_j^{e_j}$ are not too small, $w_i \neq w_j \bmod n$. In this case, $\gcd(w_i - w_j, n)$ is a non-trivial factor of n .

If we consider this algorithm modulo $q_j^{e_j}$, we see that it is a Floyd's cycle-finding algorithm in a space of size smaller than C . Consequently, cycles are searched in parallel for all the factors $q_j^{e_j}$ of n and the solution is found in average time $O(\sqrt{C})$ exponentiations to the power λ_0 modulo n using a fixed (small) amount of memory. It is important to notice that we just need that the z_i generate large subgroups modulo each factor of n and not necessarily a large subgroup of \mathbb{Z}_n^* .

Other method. A. Joux suggested a different analysis based on Pollard's $p - 1$ factoring method and that leads to the same conclusions about the choice of K .

5 Performances

Let us consider the following typical application: a prover wishes to generate a non-interactive proof of knowledge of the factorization of a 1024-bit integer n ($|n| = 1024$). In order to reach a high level of security, we choose $k = 80$, $\ell = 1$ and $B = 2^k = 2^{80}$ in order to obtain a probability of success for a dishonest prover smaller than $1/2^{80}$ (lemma 1). The choice of A is directed by the results of theorems 4, 5 and 7 on the completeness, soundness and zero-knowledge property. We have to take A much larger than $(n - \varphi(n))\ell B$ and smaller than n , e.g. $A = 2^{1024}$. Finally, the choice of $C = 2^{42}$ allows to take $K = 3$ according to theorem 8.

We can only consider integers n with less than 16 prime factors. The protocol is secure for the prover if all the factors are much more than 84-bits long, e.g. 128-bits long. For numbers with smaller factors, a dishonest prover could not cheat but a (possibly dishonest) verifier could learn non negligible information about the factorization of n . Notice that for any cryptographic application which requires composite integers to perform secure computations, we cannot reasonably assume the intractability of the factorization of an integer with prime factors shorter than 2^{128} .

With this choice of parameters, a proof requires 3 exponentiations modulo n for the prover and for the verifier. The proof is very short ($80 + 1024 = 1104$ bits long) and of about the same size than the integer n .

6 A Variant to Prove the Knowledge of the Factorization of any Integer

As we previously said, our protocol can only be used with integers n such that $(n - \varphi(n)) \times 2^k \ll n$, i.e. integers without small prime factors. Adi Shamir suggested an interesting variant to deal with such numbers:

Let a, b, ℓ and K be integers. Let z_1, \dots, z_K be K elements randomly chosen in \mathbb{Z}_n^* . A round of proof consists for the prover in randomly choosing an integer r in $[0, 2^a]$ and computing, for $i = 1..K$, the *commitments* $x_i = z_i^r \bmod n$. Then he sends the x_i s to the verifier who answers a *challenge* e randomly chosen in $[0, 2^b]$. The prover computes an answer $y \in [0, 2^a]$ such that $y = r + c \times \varphi(n) - e \times 2^a$ for a suitable value of c . He sends it to the verifier who checks $0 \leq y < 2^a$ and, for $i = 1..K$, $z_i^{y+e \times 2^a} = x_i \bmod n$. A complete proof consists in repeating ℓ times the elementary round.

This scheme is based on the ability to compute an integer y equal to r modulo $\varphi(n)$, with its b leading bits fixed to e , when $\varphi(n)$ is known. The correctness is satisfied as soon as $2^a \gg \varphi(n)$ so we impose $2^a \gg n$. The proof of soundness is similar to the proof of theorem 5. Finally, the protocol is also statistically zero-knowledge when $2^a \gg n$.

Acknowledgments

We would like to thank Adi Shamir and Antoine Joux for their helpful comments and suggestions.

References

1. M. Bellare and P. Rogaway. Random Oracles are Practical: a paradigm for designing efficient protocols. In *Proc. of the 1st CCCS*, pages 62–73. ACM press, 1993. [156](#)
2. M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge. *SIAM journal of computing*, 20(4):1084–1118, 1991. [155](#)
3. F. Boudot and J. Traoré. Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. In *Proc of ICICS'99*. Springer-Verlag, 1999. [148](#)
4. J. Boyar, K. Friedl, and C. Lund. Practical Zero-Knowledge Proofs: Giving Hints and Using Deficiencies. *Journal of Cryptology*, 4(3):185–206, 1991. [148](#)
5. J. Camenisch and M. Michels. Proving in Zero-Knowledge That a Number Is the Product of Two Safe Primes. In *Eurocrypt '99*, LNCS 1592, pages 107–122. Springer-Verlag, 1999. [148](#)
6. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology Revisited. In *Proc. of the 30th STOC*, pages 209–218. ACM Press, 1998. [156](#)
7. H. Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics 138. Springer-Verlag, 1993. [166](#)
8. U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, 1:77–95, 1988. [151](#), [156](#)
9. A. Fiat and A. Shamir. How to Prove Yourself: practical solutions of identification and signature problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, 1987. [147](#), [156](#)
10. P. Flajolet and A. Odlyzko. Random Mapping Statistics. In *Eurocrypt '89*, LNCS 434, pages 329–354. Springer-Verlag, 1990. [157](#)
11. Z. Galil, S. Haber, and M. Yung. A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems. In *Proc. of the 26th FOCS*, pages 360–371. IEEE, 1985. [148](#)
12. R. Gennaro, D. Micciancio, and T. Rabin. An Efficient Non-Interactive Statistical Zero-Knowledge Proof System for Quasi-Safe Prime Products. In *Proc. of the 5th CCCS*, pages 67–72. ACM press, 1998. [148](#)
13. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto '94*, LNCS 839, pages 202–215. Springer-Verlag, 1994. [155](#)
14. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304. ACM Press, 1985. [147](#), [148](#), [154](#)
15. L. C. Guillou and J.-J. Quisquater. A “Paradoxal” Identity-Based Signature Scheme Resulting from Zero-Knowledge. In *Crypto '88*, LNCS 403, pages 216–231. Springer-Verlag, 1989. [147](#)
16. M. Liskov and D. Silverman. A Statistical Limited-Knowledge Proof for Secure RSA Keys. Technical report, RSA Laboratories, 1998. [148](#)
17. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. [157](#)

18. G. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13:300–317, 1976. [154](#), [165](#)
19. NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBLICATION 180–1, april 1995. [155](#)
20. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, 1996. [156](#)
21. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 1999. to appear, available at <http://www.dmi.ens.fr/~pointche/>. [156](#)
22. J. M. Pollard. A Monte Carlo Methods for Factorization. *BIT*, 15:331–334, 1975. [157](#)
23. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Eurocrypt '98*, LNCS 1403, pages 422–436. Springer-Verlag, 1998. [155](#)
24. J.N. Rosser and L. Schoenfeld. Approximate Formulas for some Functions of Prime Numbers. *Illinois Journal of Mathematics*, 6(1):64–94, march 1962. [162](#)
25. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991. [147](#), [152](#)
26. M. Stadler. Publicly verifiable secret sharing. In *Eurocrypt '96*, LNCS 1070, pages 190–199. Springer-Verlag, 1996. [147](#)
27. D. R. Stinson. *Cryptography, Theory and Practice*. CRC Press, 1995. [166](#)
28. M. Tompa and H. Woll. Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information. In *Proc. of the 28rd FOCS*, pages 472–483. IEEE, 1987. [148](#)
29. J. van de Graaf and R. Peralta. A Simple and Secure Way to Show the Validity of Your Public Key. In *Crypto '87*, LNCS 293, pages 128–134. Springer-Verlag, 1988. [148](#)

A On the Generation of $\mathbb{Z}_{p^e}^*$ (Proofs)

A.1 Generation of $\mathbb{Z}_{p^e}^*$ with one Randomly Chosen Element

We first recall known properties of the Euler function:

- Fact 1** – If p is prime and $e \in \mathbb{N}^*$, then $\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p - 1)$
 – If $\gcd(m, n) = 1$ then $\varphi(m \times n) = \varphi(m) \times \varphi(n)$ (φ is said to be multiplicative)
 – If $n = q_1^{e_1} \times q_2^{e_2} \times \dots \times q_k^{e_k}$ is the prime factorization of n , then

$$\varphi(n) = n \times \left(1 - \frac{1}{q_1}\right) \left(1 - \frac{1}{q_2}\right) \dots \left(1 - \frac{1}{q_k}\right)$$

A first question about the generation of $\mathbb{Z}_{p^e}^*$ is the following: for a randomly chosen element g of $\mathbb{Z}_{p^e}^*$, what is the probability for g to be a generator of $\mathbb{Z}_{p^e}^*$, i.e. $\langle g \rangle = \mathbb{Z}_{p^e}^*$? The classical answer below allows to count the number of generators of $\mathbb{Z}_{p^e}^*$.

Fact 2 Suppose that g is a generator of $\mathbb{Z}_{p^e}^*$. Then $h = g^i \bmod p^e$ is also a generator of $\mathbb{Z}_{p^e}^*$ if and only if $\gcd(i, \varphi(p^e)) = 1$. It follows that the number of generators of $\mathbb{Z}_{p^e}^*$ is $\text{Card}(\mathbb{Z}_{\varphi(p^e)}^*) = \varphi(\varphi(p^e))$.

$$\Pr_{g \in \mathbb{Z}_{p^e}^*} \{\langle g \rangle = \mathbb{Z}_{p^e}^*\} = \frac{\varphi(p^{e-1}(p-1))}{p^{e-1}(p-1)} = \begin{cases} \varphi(p-1)/(p-1) & \text{if } e=1 \\ \varphi(p-1)/p & \text{if } e>1 \end{cases}$$

It can be noted that this probability can be small. For example, if p is such that $p - 1$ is the product of the first α prime numbers ($p - 1 = \prod_{i=1}^{\alpha} p_i$), the probability to generate \mathbb{Z}_p^* with a single element is $\varphi(p - 1)/(p - 1) = \prod_{i=1}^{\alpha} (1 - 1/p_i) \cong \frac{e^{-\gamma}}{\ln(\alpha \ln \alpha)}$ where γ is Euler's constant (see for example [24]). For $\alpha = 55$, p is 340 bits long and this probability is about 1/10. This means that we need to try more than 500 elements in order to find a generator with probability very close to 1 such as $1 - 1/2^{80}$.

More precisely, the following fact proved in [24] allows to lower bound the probability of fact 2:

Fact 3 *For all integers $x \geq 5$, $\frac{\varphi(x)}{x} > \frac{1}{6 \ln \ln x}$ but there does not exist any constant C such that $\varphi(x) > C \times x$ for any integer x .*

Theorem 1 *For any prime number $p \geq 7$, for any $e \geq 1$,*

$$\Pr_{g \in \mathbb{Z}_{p^e}^*} \{ \langle g \rangle = \mathbb{Z}_{p^e}^* \} = \frac{\varphi(\varphi(p^e))}{\varphi(p^e)} > \frac{1}{7 \ln \ln p}$$

Proof. The probability for a randomly chosen element g of $\mathbb{Z}_{p^e}^*$ to generate the all group is greater than $\varphi(p - 1)/p$ so, using the previous fact, if $p \geq 7$ the probability is larger than $(1 - 1/p) \times 1/(6 \ln \ln(p - 1)) > 1/(7 \ln \ln p)$. \square

A.2 Generation of $\mathbb{Z}_{p^e}^*$ with K Randomly Chosen Elements

A natural question is how the probability of fact 2 is modified if we choose K elements g_1, \dots, g_K of $\mathbb{Z}_{p^e}^*$ instead of one. To answer this problem, we first generalize the Euler totient function and define φ_K for all integers $K \geq 1$ by:

- If p is prime and $e \geq 1$, then $\varphi_K(p^e) = p^{Ke} - p^{K(e-1)}$
- If $\gcd(m, n) = 1$ then $\varphi_K(m \times n) = \varphi_K(m) \times \varphi_K(n)$ (implies $\varphi_K(1) = 1$)

We note that for $K = 1$, $\varphi_1 = \varphi$.

Lemma 2 *If $n = q_1^{e_1} \times q_2^{e_2} \times \dots \times q_k^{e_k}$ is the prime factorization of n , then*

$$\varphi_K(n) = n^K \times \left(1 - \frac{1}{q_1^K}\right) \left(1 - \frac{1}{q_2^K}\right) \dots \left(1 - \frac{1}{q_k^K}\right)$$

The functions φ_K allow to generalize fact 2 to the case of K generators:

Lemma 3 *The number of K -tuples of $(\mathbb{Z}_{p^e}^*)^K$ that generate $\mathbb{Z}_{p^e}^*$ is $\varphi_K(\varphi(p^e))$.*

Proof. Let (g_1, \dots, g_K) be a K -tuple of $(\mathbb{Z}_{p^e}^*)^K$. Let g be a generator of $\mathbb{Z}_{p^e}^*$; for $i = 1, \dots, K$, we define $\alpha_i \in \mathbb{Z}_{\varphi(p^e)}$ by the relation $g^{\alpha_i} = g_i \bmod p^e$.

We first notice that (g_1, \dots, g_K) generates $\mathbb{Z}_{p^e}^*$ if and only if the ideal generated by $\alpha_1, \alpha_2, \dots, \alpha_K$ in the ring $\mathbb{Z}_{\varphi(p^e)}$ is the entire ring. Bezout equality shows that this occurs iff $\gcd(\alpha_1, \dots, \alpha_K, \varphi(p^e)) = 1$.

Let us count the number of K -tuples $(\alpha_1, \dots, \alpha_K) \in (\mathbb{Z}_{\varphi(p^e)})^K$ such that $\gcd(\alpha_1, \dots, \alpha_K, \varphi(p^e)) = 1$. Let $\prod_{i=1}^t q_i^{f_i}$ be the prime factorization of $\varphi(p^e)$. We know that

$$\gcd(x, \prod_{i=1}^t q_i^{f_i}) = 1 \Leftrightarrow \forall i \leq t \quad \gcd(x, q_i^{f_i}) = 1 \Leftrightarrow \forall i \leq t \quad \gcd(x \bmod q_i^{f_i}, q_i^{f_i}) = 1$$

Using the Chinese remainder theorem, the problem reduces to counting the number of K -tuples $(\beta_1, \dots, \beta_K)$ of $(\mathbb{Z}_{q_i^{f_i}})^K$ such that $\gcd(\beta_1 \bmod q_1^{f_1}, \dots, \beta_K \bmod q_K^{f_K}, q_i^{f_i}) = 1$ for $i = 1, \dots, t$. The K -tuples that do not verify this relation for a fixed index i are of the form $(q_i \gamma_1, \dots, q_i \gamma_K)$ where $(\gamma_1, \dots, \gamma_K) \in (\mathbb{Z}_{q_i^{f_i-1}})^K$ and there are exactly $q_i^{K(f_i-1)}$ such K -tuples.

Finally there are $\prod_{i=1}^t q_i^{K f_i} - q_i^{K(f_i-1)}$ K -tuples of $(\mathbb{Z}_{\varphi(p^e)})^K$ such that $\gcd(\alpha_1, \dots, \alpha_K, \varphi(p^e)) = 1$ and this is equal to $\prod_{i=1}^t \varphi_K(q_i^{f_i}) = \varphi_K(\varphi(p^e))$ since φ_K is multiplicative. \square

Theorem 2 For any odd prime number p , for any $e \geq 1$, for any $K \geq 2$,

$$\text{with } \mathcal{P} = \Pr_{\{(g_i)_{i \in [1, K]} \in (\mathbb{Z}_{p^e})^K\}} \left\{ \left\langle (g_i)_{i \in [1, K]} \right\rangle = \mathbb{Z}_{p^e}^* \right\}$$

$$\mathcal{P} = \frac{\varphi_K(\varphi(p^e))}{\varphi(p^e)^K} > \frac{1}{\zeta(K)} > 1 - \frac{K+1}{K-1} \times \frac{1}{2^K}$$

Proof: Let us first introduce a notation: for any integer x , let S_x be the set of the indices i such that p_i is a factor of x .

From the previous lemma, we know that the probability for a K -tuple of $(\mathbb{Z}_{p^e})^K$ to generate $\mathbb{Z}_{p^e}^*$ is $P = \frac{\varphi_K(\varphi(p^e))}{\varphi(p^e)^K}$. Lemma 2 shows that P is equal to the product $\prod_{i \in S_{\varphi(p^e)}} 1 - \frac{1}{p_i^K}$. The inverse of each term $1 - 1/p_i^K$ can be expanded in power series: $(1 - 1/p_i^K)^{-1} = \sum_{j=0}^{+\infty} (1/p_i^K)^j$. The probability P is a product of series with positive terms, $P = \left(\prod_{i \in S_{\varphi(p^e)}} \sum_{\alpha_i=0}^{+\infty} \frac{1}{p_i^{\alpha_i \times K}} \right)^{-1}$ so we can distribute terms and obtain that P^{-1} is the sum of $1/d^K$ where d ranges over integers whose prime factors are among p_i s, $i \in S_{\varphi(p^e)}$. This sum is smaller than the unrestricted sum $\sum_{d=1}^{+\infty} 1/d^K = \zeta(K)$. Finally, we obtain $P > 1/\zeta(K)$.

The Riemann Zeta function is bounded by the following integral: $\zeta(K) = \sum_{d=1}^{+\infty} 1/d^K < 1 + 1/2^K + \int_2^{+\infty} dx/x^K = 1 + \frac{K+1}{K-1} \times \frac{1}{2^K}$. Since for all $x > -1$, $1/(1+x) \geq 1-x$, $1/\zeta(K) > 1 - \frac{K+1}{K-1} \times \frac{1}{2^K}$. \square

This result provides a lower bound independent of p and e . This is quite surprising since for $K = 1$, theorem 1 proves that such a non-zero bound does not exist.

A.3 Generation of a Large Subgroup of $\mathbb{Z}_{p^e}^*$ with K Randomly Chosen Elements

Another statement of theorem 2 is that we need to randomly choose K elements in $\mathbb{Z}_{p^e}^*$ in order to generate the group with probability greater than $1 - 1/2^K$. For a probability very close to one such that $1 - 1/2^{80}$, K becomes quite large even if experiments show that a few randomly chosen elements always generate very large subgroups of $\mathbb{Z}_{p^e}^*$. We now make this observation precise by establishing a lower bound of the probability $\mathcal{P}_K^C(p^e)$ that K elements generate a subgroup of size greater than $\text{Card}(\mathbb{Z}_{p^e}^*)/C$. We first generalize lemma 3 for subgroups of $\mathbb{Z}_{p^e}^*$.

Lemma 4 *For any divisor d of $\varphi(p^e)$, the number of K -tuples of $(\mathbb{Z}_{p^e}^*)^K$ that generate a subgroup of $\mathbb{Z}_{p^e}^*$ of order d is $\varphi_K(d)$.*

Proof: Let (g_1, \dots, g_K) be a K -tuple of $(\mathbb{Z}_{p^e}^*)^K$ and g be a generator of $\mathbb{Z}_{p^e}^*$; for $i = 1, \dots, K$ we define $\alpha_i \in \mathbb{Z}_{\varphi(p^e)}$ by the relation $g^{\alpha_i} \equiv g_i \pmod{p^e}$. The K -tuple (g_1, \dots, g_K) generates a subgroup of $\mathbb{Z}_{p^e}^*$ of order d if and only if the size of the ideal generated by the α_i s in the ring $\mathbb{Z}_{\varphi(p^e)}$ is d . Bezout equality shows that this is equivalent to $\gcd(\alpha_1, \dots, \alpha_K, \varphi(p^e)) = \varphi(p^e)/d$.

Let us factor $\varphi(p^e)$ and d : $\varphi(p^e) = \prod_{i=1}^t q_i^{f_i}$ and $d = \prod_{i=1}^t q_i^{d_i}$ with $f_i \geq d_i \geq 0$. The number of K -tuples $(\beta_1, \dots, \beta_K)$ of $(\mathbb{Z}_{q_i^{f_i}})^K$ such that $\gcd(\beta_1, \dots, \beta_K, q_i^{f_i}) = q_i^{f_i-d_i}$ is equal to the number of K -tuples of $(\mathbb{Z}_{q_i^{d_i}})^K$ whose gcd with $q_i^{d_i}$ is 1, i.e. $\varphi_K(q_i^{d_i})$. The total number of K -tuples of $(\mathbb{Z}_{p^e}^*)^K$ that generate a subgroup of $\mathbb{Z}_{p^e}^*$ of order $\varphi(p^e)/d$ is consequently $\prod_{i=1}^t \varphi_K(q_i^{d_i}) = \varphi_K(d)$. \square

Let K and C be two integers ≥ 1 ; we note

$$\mathcal{P}_K^C(n) = \Pr_{\{(g_i)\}_{i \in [1, K]} \in (\mathbb{Z}_n^*)^K} \left\{ \text{Card} \left\langle (g_i)_{i \in [1, K]} \right\rangle \geq \frac{\text{Card}(\mathbb{Z}_n^*)}{C} \right\}$$

Theorem 3 *For any $C \geq 1$ and $K \geq 2$,*

$$\mathcal{P}_K^C(p^e) > \frac{1}{\zeta(K)} \times \sum_{d=1}^C \frac{1}{d^K} > 1 - \frac{1}{(K-1)C^{K-1}\zeta(K)}$$

Proof: The probability $\mathcal{P}_K^C(p^e)$ is

$$\sum_{\delta \geq \varphi(p^e)/C} \left(\Pr_{\{(g_i)\}_{i \in [1, K]} \in (\mathbb{Z}_n^*)^K} \left\{ \text{Card} \left\langle (g_i)_{i \in [1, K]} \right\rangle = \delta \right\} \right)$$

Lemma 4 shows that this sum is equal to $\sum_{\delta | \varphi(p^e), \delta \geq \varphi(p^e)/C} \frac{\varphi_K(\delta)}{\varphi(p^e)^K}$ and this expression can also be written $\sum_{d_0 | \varphi(p^e), d_0 \leq C} \frac{1}{d_0^K} \times \frac{\varphi_K(\varphi(p^e)/d_0)}{\varphi(p^e)/d_0^K}$ if we replace δ by $\varphi(p^e)/d_0$.

Let us calculate

$$\mathcal{P}_K^C(p^e) \times \zeta(K) = \sum_{d_0|\varphi(p^e), d_0 \leq C} \frac{1}{d_0^K} \times \prod_{i \in S_{\varphi(p^e)/d_0}} \left(1 - \frac{1}{p_i^K}\right) \times \zeta(K)$$

In the Riemann Zeta function $\zeta(K) = \sum_{\beta=1}^{+\infty} 1/\beta^K$ the index β can be written as the product of two integers, d_1 which is relatively prime with $\varphi(p^e)/d_0$ and d_2 whose factors are among the p_i s for $i \in S_{\varphi(p^e)/d_0}$. As in the proof of theorem 2, we note that the sum $\sum 1/d_2^K$ is equal to the inverse of $\prod_{i \in S_{\varphi(p^e)/d_0}} \left(1 - \frac{1}{p_i^K}\right)$ so we obtain that $\mathcal{P}_K^C(p^e) \times \zeta(K) = \sum_{d_0|\varphi(p^e), d_0 \leq C} \left(\frac{1}{d_0^K} \times \sum_{\gcd(d_1, \varphi(p^e))=1} \frac{1}{d_1^K}\right)$. Finally let us observe that all the integers smaller than C can be uniquely decomposed in the product of a divisor d_0 of $\varphi(p^e)$ smaller than C and of an integer d_1 relatively prime with $\varphi(p^e)$. As a consequence, $\mathcal{P}_K^C(p^e) \times \zeta(K)$ is greater than $\sum_{d=1}^C 1/d^K$.

The end of the proof is a consequence of calculus techniques for comparing integrals and series:

$$\begin{aligned} \mathcal{P}_K^C(p^e) &> \frac{\zeta(K) - \sum_{d=C+1}^{+\infty} \frac{1}{d^K}}{\zeta(K)} \\ &> 1 - \frac{1}{\zeta(K)} \times \int_C^{+\infty} \frac{dx}{x^K} = 1 - \frac{1}{(K-1)C^{K-1}\zeta(K)} \end{aligned}$$

□

B Miller's Factoring Algorithm

Let n be an integer whose factorization has to be found and $L = 2^s \times r$, with r an odd integer, a multiple of $\lambda(n)$. We can first assume that n is odd because if L is a multiple of $\lambda(2^\alpha n) = \text{lcm}(\lambda(2^\alpha), \lambda(n))$ it is still a multiple of $\lambda(n)$. The following algorithm is due to Miller [18]:

Algorithm $\text{Fact}(n, L)$

1. choose w at random in $[1, n - 1]$
2. if $1 < \gcd(w, n) < n$ then return $\gcd(w, n)$
3. compute $v = w^r \bmod n$
4. if $v = 1 \bmod n$ then return **Fail**
5. while $v \neq 1 \bmod n$ do $v_0 = v$ and $v = v^2 \bmod n$
6. if $v_0 = -1 \bmod n$ then return **Fail** else return $\gcd(v_0 + 1, n)$

This algorithm allows to find a non-trivial factor of n , i.e. a factor different from 1 and n . It can be recursively used to completely factor n .

Lemma 5 Let n be a k -bit integer and $L < X$ be a multiple of $\lambda(n)$. Then $\text{Fact}(n, L)$ outputs the factorization of n in expected time $O(\eta \times \log X)$ modular multiplications, where η is the number of distinct prime factors of n .

Proof: Let $\prod_{i=1}^{\eta} p_i^{e_i}$ be the prime factorization of n . We first prove that the algorithm $\text{Fact}(n, L)$ returns a non-obvious factor of n with probability $> 1 - 1/2^\eta$, after at most $O(\log X)$ arithmetical operations. The underlying idea is the same as in the Rabin-Miller primality test; if w is an element of \mathbb{Z}_n^* , $w^L = 1 \pmod{n}$ so if we find α such that $w^{2^\alpha r} \neq \pm 1$ and $w^{2^{\alpha+1}r} = 1$, we obtain $(w^{2^\alpha r} + 1)(w^{2^\alpha r} - 1) = 0 \pmod{n}$ and thus $\gcd(n, w^{2^\alpha r} + 1)$ is a non-trivial factor of n .

The proof generalizes the one presented in [27, chapter 4] when n is an RSA modulus. We first need the following notations in order to analyze the algorithm: $\lambda(p_i^{e_i}) = \varphi(p_i^{e_i}) = p_i^{e_i-1}(p_i - 1) = 2^{\alpha_i} p'_i$ with p'_i odd, $\mathcal{P} = \prod_{i=1}^{\eta} p'_i$, $\beta = \min\{\alpha_i, 1 \leq i \leq \eta\}$, g_i is a generator of the cyclic group $\mathbb{Z}_{p_i^{e_i}}^*$, u_i is such that $w = g^{u_i} \pmod{p_i^{e_i}}$ and $0 \leq u_i < \lambda(p_i^{e_i})$.

Let us count the number of w for which the algorithm fails. This happens if $w^r = 1 \pmod{n}$ or $w^{2^t r} = -1 \pmod{n}$ for an integer t in $[0, s]$. Because of the Chinese remainder theorem, $w^r = 1 \pmod{n}$ is equivalent to $\forall i, w^r = 1 \pmod{p_i^{e_i}}$. It can be seen that $w^r = 1 \pmod{p_i^{e_i}}$ if and only if there exists μ_i in $[0, p'_i]$ such that $u_i = \mu_i 2^{\alpha_i}$. So $w^r = 1 \pmod{n}$ has $\prod_{i=1}^{\eta} p'_i = \mathcal{P}$ solutions.

Using same ideas, $w^{2^t r} = -1 \pmod{n}$ is equivalent to $w^{2^t r} = -1 \pmod{p_i^{e_i}}$ for all $i = 1..n$. The latter equation has no solution if $t \geq \alpha_i$ and it has $\lambda(p_i)/(2 \times 2^{\alpha_i-t-1}) = 2^t p'_i$ solutions if $t < \alpha_i$. So $w^{2^t r} = -1 \pmod{p_i^{e_i}}$ as no solution if $t \geq \beta$ and $2^{\eta t} \mathcal{P}$ solutions if $t < \beta$. Finally, the number of values w for which the algorithm fails is less than $\mathcal{P} + \sum_{t=0}^{\beta-1} 2^{\eta t} \mathcal{P}$. We can easily prove that this is less than $n/2^{\eta-1}$. The algorithm succeeds with probability $> 1 - 1/2^{\eta-1}$ and performs less than $s + 1 \leq \log_2 X \leq \log_2 L$ modular multiplications. If we use the algorithm until we find a non trivial factor of n , the expected number of executions is smaller than $1/(1 - 1/2^{\eta-1}) \leq 2$.

Fact can be used to recursively factor n . With this aim, we also need a prime power detection algorithm such as the one proposed in [7, page 41] and which is also based on Rabin-Miller ideas. If we want to factor an integer n , we first test if n is a prime power and if not we call $\text{Fact}(n, L)$ as long as it fails. After about two tries, we obtain $n = n' \times n''$ and we recursively call the factorization procedure. Notice that a multiple L of $\lambda(n)$ is also a multiple of $\lambda(n')$ if n' is a factor of n . The proposed algorithm needs on average less than 2η calls to Fact so finally we can factor n with $O(\eta \times \log X)$ modular multiplications. \square

Secure and Practical Tree-Structure Signature Schemes Based on Discrete Logarithms

X.Y. Wang^{1,2}, L.C. Hui¹, K.P. Chow¹, W.W. Tsang¹, C.F. Chong¹, and H.W. Chan¹

¹ Department of Computer Science and Information Systems

The University of Hong Kong

Pokfulam, Hong Kong

² Department of Mathematics

Shandong University

Jinan 250100 PRC

hui@csis.hku.hk

Abstract. In this paper, we present another tree-structure signature scheme based on discrete logarithm problem modulo p , where p is a large prime. The basic signing algorithm is the original ElGamal signature scheme. The scheme attains ideal security, i.e., finding existential forgeries under adaptively chosen message attacks is equivalent to solving the discrete logarithm of any random integer $y \in Z_p^*$. The scheme is also efficient, it can be implemented almost as efficiently as the original ElGamal signature scheme. We can regard the scheme as an application of ElGamal signature scheme in tree-structure signature schemes.

1 Introduction

Digital signature schemes are usually based on difficult number-theoretic problems. For example, the original RSA signature scheme [20] is based on the difficulty of integer factorization, and the ElGamal signature scheme [7] is based on the difficulty of discrete logarithms. The factoring problem and discrete logarithm problem are two important problems widely used to construct various signature schemes, for example, [1, 2, 4, 5, 6, 11, 14, 15, 16, 19, 21].

A strong requirement for digital signature scheme is that the scheme has to be "existentially unforgeable under an adaptively chosen message attack" [11], or simply "existentially unforgeable". In this attack, the adversary (or the signature forger) gets to see a signature on any message of its choice, in an adaptive manner. The forger has then to produce a signature on one message that was not previously signed, without the cooperation of the signature algorithm. A signature scheme is existentially forgeable if it cannot prevent the forger in forging a signature. On the other hand, a signature scheme is existentially unforgeable if it can be proved that the scheme prevents forging a signature in the above-mentioned manner.

Many signature schemes are not existentially unforgeable. For example, both the ElGamal scheme and the RSA scheme are known to be existentially forgeable.

One practical solution to this problem is to use those schemes together with some secure hash function. That is, the message to be signed has to be processed by a hash function first, and then the hash value is signed. While this approach works fine in practice, in theory this approach has a drawback that the security of the overall signature scheme has to be depending on the security of the hash function as well. Therefore, it is interesting to design signature schemes that are existentially unforgeable even if no hash functions are used.

One approach to design existentially unforgeable signature schemes is to use a tree structure [4,5,6,11,13]. In a tree-structure signature scheme, a digital signature corresponds to a path of a specially designed tree, from the root to a leaf. Each edge in the path corresponds to some "basic" digital signature technique, for example RSA. A signature in a tree-structure signature scheme can be viewed as a sequence of signatures in the basic scheme together with some extra information.

Early tree-structure signatures usually have larger signature sizes than similar signatures directly based on the respective basic problems. For example, the size of [13] is $O(l^2 \log i)$.

[11] reduces the size to $O(ld)$, where d is the depth of a binary tree used in the signature scheme, and l is a security parameter which often indicates the length of the main input, for example, in [11], l is the binary length of the modulo $n = pq$. [5] and [6] also have small signature size of $O(ld)$, here d also is the depth of the tree and not more than 3. This is only a constant times the signature size of the basic algorithm, which is the original RSA signature scheme [20]. Tree-structure signature schemes having this efficiency property are known as "practical" schemes.

In the security aspect of [11], there are no existential forgery under the assumption that the factoring problem is difficult to solve. This means that if somebody can forge a signature of [11], then there is a probabilistic polynomial-time attack with non-negligible probability to factor the modulo n . In this paper, we denote the scheme as achieving "ideal" security under the factorization assumption. In the security aspect of [5] and [6], two schemes achieve the "ideal" security under the RSA assumption.

[4] describes a general method for constructing secure tree-structure signature schemes based on interactive protocols. In addition, the paper introduces an example of applying the general method to obtain a secure signature scheme based on the discrete logarithm problem. The scheme achieves the "ideal" security under the discrete logarithm assumption.

In this paper, we propose an alternate tree-structure signature scheme also based on discrete logarithms. Our scheme uses the original ElGamal signature [7] as its basic signing algorithm. The scheme attains ideal security under the discrete logarithm assumption, i.e., finding existential forgeries under adaptively chosen message attacks is equivalent to solving the discrete logarithm of any random integer $y \in Z_p^*$. The scheme has similar efficiency as [4]. It outputs a signature with a signature size about $2d$ times that of the original ElGamal signature size, here d is the depth of the binary tree used in the scheme. For

example, If the depth of the tree is 32, the signer at most can generate 2^{32} signatures, then the signature size is 64 times that of the original ELGamal signature. Therefore our scheme is also practical.

The organization of the rest of the paper is as follows. Section 2 gives an overview of tree-structure signature schemes. Section 3.1 describes the details of the scheme, and sections 3.2 and 3.3 analyse the efficiency and security of the scheme, respectively. Finally in section 4, the paper is concluded.

2 Tree-Structure Signatures

The tree structure in this paper is a binary tree which is similar to that of the tree-structure signature scheme in [11]. d represents the depth of the tree. A signer can make at most 2^d signatures.

A value is associated with each internal node. We denote the value in the root by $Y_0 = (x_0, y_{01}, y_{02})$. The signing process starts from the leftmost unused path. The value in the root is a part of the public key. The values in all internal nodes are progressively defined with the signing process by the signer. For example, If the signer generates the i -th signature, all the internal nodes appeared in the previous $(i - 1)$ signatures will have their values defined.

Assume that the i -th path is (i_1, i_2, \dots, i_d) which has the same first j internal nodes with $(i - 1)$ -th path, then in the i -th path, the first j nodes have their values defined coincide with those of the $(i - 1)$ -th signature. The remaining $d - j$ nodes will be given their new values according to the signing algorithm. All the nodes that do not appear in first i signatures will not have their values defined. After 2^d different signatures have been generated, any node of the tree employs a fixed value.

In the signature scheme presented in this paper, each internal node associates with a sequence of 3 elements. The choice of the value corresponding to each leaf is different from the values of internal nodes. The value of each leaf is a sequence containing two elements.

In addition, we provide that if the signer generates the i -th signature, he must store the $(i - 1)$ -th signature in advance. This is a property of tree-structure signatures. Certainly, the scheme can be made totally "memoryless" like [11].

3 Signature Schemes Based on Discrete Logarithm Problem

3.1 The Description of the Signature Scheme

Key generation: The signer selects a big prime p such that $p - 1 = aq$, q is a large prime and a is a small factor. Select a generator $g \in Z_p^*$. Select $g_1 \equiv g^{\gamma_1} \pmod{p}$, $g_2 \equiv g^{\gamma_2} \pmod{p}$, $x_0 \equiv g^{\alpha_0} \pmod{p}$, $y_{0t} \equiv g^{u_{0t}} \pmod{p}$, $t = 1, 2$ such that $g.c.d(\alpha_0, p - 1) = 1$, $g.c.d(\gamma_t, p - 1) = 1$, $g.c.d(u_{0t}, p - 1) = 1$, $t = 1, 2$, and $\alpha_0, \gamma_t, u_{0t}$ are random integers in Z_{p-1}^* .

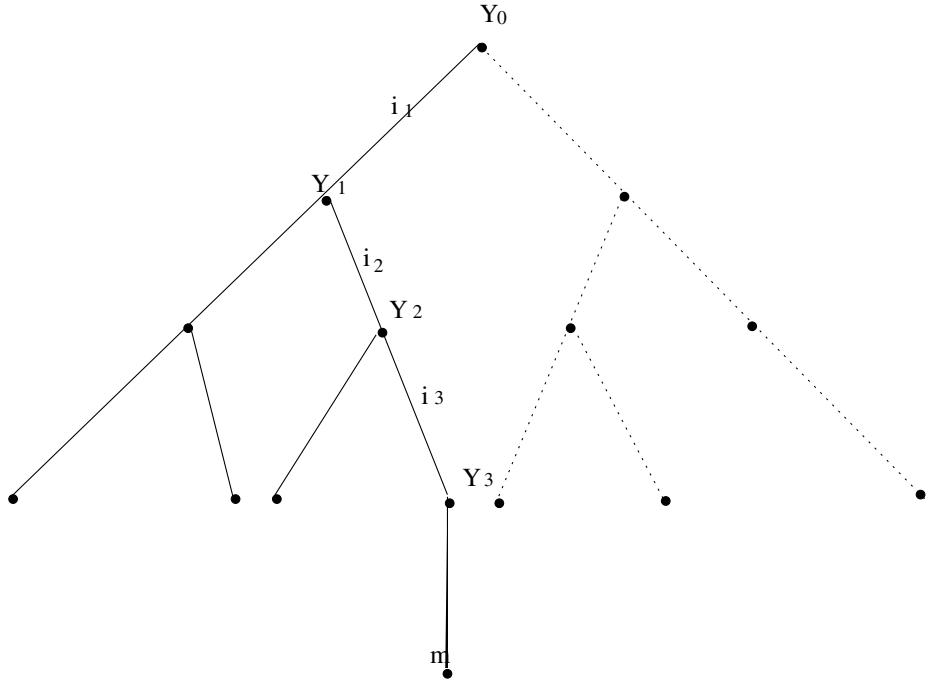


Fig. 1. The Tree with $d = 3$

Public key: The public key is (p, Y_0, g, g_1, g_2) , where Y_0 is a sequence including 3 elements $Y_0 = (x_0, y_{01}, y_{02})$.

Private key: The private key is $(\alpha_0, \gamma_1, \gamma_2, u_{01}, u_{02})$.

Signing: In the tree of the scheme, each node will be associated with a sequence of 3 elements. We denote the sequence as $Y_j = (x_j, y_{j1}, y_{j2})$, where x_j, y_{j1}, y_{j2} are selected randomly by the signer, $\text{index}_{p,g}(x_j), \text{index}_{p,g}(y_{jt}), t = 1, 2$ are kept secret by the signer until the next signature is generated. $\text{index}_{p,g}(x_j)$ denotes the discrete logarithm of x_j about the modulo p and the base g .

The signing can be described as follows (see Figure 1):

Step 1: The signer chooses the i -th path (i_1, i_2, \dots, i_d) of the tree for the i -th signature. If the nodes i_1, i_2, \dots, i_d have their corresponding sequences defined in the first $(i - 1)$ signatures, let

$$Y_k \leftarrow Y_k, k = 1, 2, \dots, j$$

$$s_k \leftarrow s_k, r_k \leftarrow r_k, t_k \leftarrow t_k, k = 1, 2, \dots, j$$

Otherwise choose random $\alpha_k, u_{kt} \in Z_{p-1}^*, k = j + 1, \dots, d - 1, t = 1, 2$, where $\text{g.c.d}(\alpha_k, p - 1) = 1$ and $\text{g.c.d}(u_{kt}, p - 1) = 1$, and let

$$x_k \leftarrow g^{\alpha_k} \bmod p$$

$$y_{kt} \leftarrow g^{u_{kt}} \bmod p$$

$$Y_k \leftarrow (x_k, y_{k1}, y_{k2})$$

Step 2: For $k = d$, choose random $\alpha_d, u_{d1} \in Z_{p-1}^*$ such that $g.c.d(\alpha_d, p-1) = 1$, and $g.c.d(u_{d1}, p-1) = 1$. Let

$$x_d \leftarrow g^{\alpha_d} \bmod p$$

$$y_{d1} \leftarrow g^{u_{d1}} \bmod p$$

$$Y_d \leftarrow (x_d, y_{d1})$$

Step 3: For $k = j+1, \dots, d$, if x_k is the i_k -th child of its parent, here $i_k = 1$ or 2. Compute s_k, t_k, r_k such that:

$$g^{x_k} \equiv x_{k-1}^{s_k} \cdot (y_{k-1, i_k})^{x_{k-1}} \bmod p \quad (1')$$

$$g_1^{y_{k1}} g_2^{y_{k2}} \equiv x_k^{t_k} \cdot x_0^{r_k} \cdot g^{x_k} \bmod p$$

If $k = d$, let $y_{d2} = 0$.

This is equivalent to computing s_k, t_k, r_k such that:

$$x_k \equiv \alpha_{k-1} \cdot s_k + u_{k-1, i_k} \cdot x_{k-1} \bmod (p-1)$$

$$\gamma_1 \cdot y_{k1} + \gamma_2 \cdot y_{k2} \equiv \alpha_k \cdot t_k + \alpha_0 \cdot r_k + x_k \bmod (p-1)$$

(Note: The second equation has many solutions, the signer randomly selects one.)

Step 4: For the message $m \in [0, q-1]$, compute

$$g^m \equiv x_d^s \cdot (y_{d1})^{x_d} \bmod p$$

Output: The signature of message m is:

$$S(m) = (Y_1, i_1, s_1, t_1, r_1; \dots, Y_d, i_d, s_d, t_d, r_d; m, s)$$

here $Y_d = (x_d, y_{d1})$.

Verification: Given the signature $S(m)$, the verification of a signature is as follows:

$$S(m) = (Y_1, i_1, s_1, t_1, r_1; \dots, Y_d, i_d, s_d, t_d, r_d; m, s)$$

The receiver verifies that:

$$g^{x_k} \equiv x_{k-1}^{s_k} \cdot (y_{k-1, i_k})^{x_{k-1}} \bmod p, \quad k = 1, 2$$

$$g_1^{y_{k1}} \cdot g_2^{y_{k2}} \equiv x_k^{t_k} \cdot x_0^{r_k} \cdot g^{x_k} \bmod p$$

$$g^m \equiv x_d^s \cdot (y_d^{x_d}) \bmod p$$

It is noted that, in the i -th signature, the sequences corresponding to all the nodes are the following:

$$(x_1, y_{11}, y_{12}), \dots, (x_{d-1}, y_{d-1,1}, y_{d-1,2}), (x_d, y_{d1})$$

s_k is equivalent to the signature of x_k , and (t_k, r_k) is the common signature of y_{k1} and y_{k2} , $k=1,2,\dots,d$. The signature $S(m)$ in fact includes all the signatures of all the elements appeared in the path.

We provide that if $S(m)$ and $S(m')$ are two different signatures of i -th path, $S(m)$ must at least include a parameter pair (x_k, y_{k1}, y_{k2}) different from the corresponding pair (x'_k, y'_{k1}, y'_{k2}) of $S(m')$ or $(x_k, y_{k1}, y_{k2}) = (x'_k, y'_{k1}, y'_{k2})$, $k = 1, 2, \dots, d$, but $m \neq m'$.

3.2 Performance of the Signature Scheme

A signer can use the tree to make at most 2^d signatures. The signature size is about $O(ld)$ bits. the public key size is about $O(l)$ bits.

When the signer generates the i -th signature, he must store the $(i - 1)$ -th, i -th signatures and the discrete logarithms of $x_k, y_{kt}, t = 1, 2$ appears in two signatures. So the storage size including the secret key is at most $O(ld)$ bits which is about a few times that of the signature size.

If the signer use a pseudorandom generator to produce pseudorandom numbers as the discrete logarithms of x_k, y_{kt} , the scheme is no "memory", i.e. the signer can makes the i -th signature without storing any information about the $i - 1$ -th signature. The details can refer to [10].

The verification is almost as efficient as the original signature scheme, and the execution time is about d times that of the original ElGamal signature verification.

3.3 The Proof of the Security about Signature Scheme

In this section, we show that the signature scheme has no existential forgery under adaptively chosen message attacks, using the simulated signature concept used in [5,6,11]. The proof is divided into two parts. In the first part (Theorem 1), we demonstrate that there is a simulator that can simulate the signing process by selecting another public key Y_0 where discrete logarithm of each element in Y_0 is unknown. The simulated signatures and the real signatures are indistinguishable. This demonstrates that our design is zero-knowledge, i.e, the real signatures cannot leave any information about secret keys. The second part (Theorem 2) shows that, if an attacker can forge a signature of the real signature scheme under an adaptively chosen message attack with a non-negligible probability, the attacker can find a probabilistic polynomial algorithm to compute the discrete logarithm of any random integer y in Z_p^* . So, our signature scheme achieves the ideal security.

Theorem 1 There is a simulator which can simulate the signing process, i.e, the simulator can select another root sequence $Y_0 = (x_0, y_{01}, y_{02})$, where $\text{index}_{p,g}(x_0)$ and $\text{index}_{p,g}(y_{0t}), t = 1, 2$ are all unknown. The simulating process and the real signing process are indistinguishable in polynomial time.

Proof: The simulator first chooses $Y_0 = (x_0, y_{01}, y_{02})$ as follows:

Select any random integer $y \in Z_p^*$, random integer $\alpha_0, \beta_0 \in Z_{p-1}^*$, Let $x_0 \leftarrow g^{\alpha_0} \cdot y^{\beta_0}$, and provide that $\text{g.c.d}(x_0, p - 1) = 1$, $\text{g.c.d}(\beta_0, p - 1) = 1$.

Select random integer $\gamma_1, \gamma_2, \delta_1, \delta_2 \in Z_{p-1}^*$ such that:

$$g_1 \leftarrow g^{\gamma_1} \cdot y^{\delta_1}$$

$$g_2 \leftarrow g^{\gamma_2} \cdot y^{\delta_2}$$

Select first elements of Y_{11}, Y_{12} which are two children of Y_0 : $x_{1t} \equiv g^{\alpha_{1t}} \cdot y^{\beta_{1t}}$, $t = 1, 2$ such that $\alpha_0 \cdot \beta_{1t} - \beta_0 \cdot \alpha_{1t} \neq 0 \pmod{p-1}$, and α_{1t} and β_{1t} are random integers in Z_{p-1}^* . Corresponding to x_{1t} , select random integer $s_{1t} \in Z_{p-1}$.

After $x_{1t}, s_{1t}, t = 1, 2$ are selected, we can select $y_{0t}, t = 1, 2$. Assume that: $y_{0t} = g^{u_{0t}} \cdot y^{v_{0t}} \pmod{p}$, $t = 1, 2$, here $u_{0t}, v_{0t}, t = 1, 2$ are unknown parameters.

Compute y_{0t} satisfies the following equation:

$$g^{x_{1t}} \equiv x_0^{s_{1t}} \cdot y_{0t}^{x_0} \pmod{p}$$

So,

$$g^{x_{1t}} \equiv g^{(\alpha_0 \cdot s_{1t} + u_{0t} \cdot x_0)} \cdot y^{\beta_0 \cdot s_{1t} + v_{0t} \cdot x_0}$$

From $\text{g.c.d}(x_0, p-1) = 1$, we know that the equation has one solution. We determine u_{0t} and v_{0t} as the only solution of the system equations:

$$\begin{cases} \alpha_0 \cdot s_{1t} + u_{0t} \cdot x_0 \equiv x_{1t} \pmod{p-1} \\ \beta_0 \cdot s_{1t} + v_{0t} \cdot x_0 \equiv 0 \pmod{p-1} \end{cases}$$

Similar to the above method, if the first element x_k of $Y_k = (x_k, y_{k1}, y_{k2})$ is defined, the other elements y_{k1} and y_{k2} will be defined after two first elements $x_{k+1,1}$ and $x_{k+1,2}$ of $Y_{k+1,1}$ and $Y_{k+1,2}$ defined where Y_{k1} and Y_{k2} are two children of Y_k .

If the simulator has output $(i-1)$ simulated signatures, for any unsigned message m , the i -th signatures $S(m)$ is generated as follows:

The simulator chooses the i -th path (i_1, i_2, \dots, i_l) of the tree. If the path has $j-1$ common internal nodes with the $(i-1)$ -th signature, the first $j-1$ nodes of the i -th path has the same sequences $(Y_1, Y_2, \dots, Y_{j-1})$ with the former signature. The sequence corresponding to j -th node only has x_j defined, and $y_{jt}, t = 1, 2$ are not defined. Similar to the choices of x_{1t}, s_{1t} and $y_{0t}, t = 1, 2$, the simulator choose $x_{j+1,t}, s_{j+1,t}, y_{jt}, t = 1, 2$ where $x_{j+1,t} \equiv g^{\alpha_{j+1,t}} \cdot y^{\beta_{j+1,t}}$, $y_{jt} \equiv g^{u_{jt}} \cdot y^{v_{jt}} \pmod{p}$, $t = 1, 2$ such that:

$$\text{g.c.d}(x_{j+1,t}, p-1) = 1$$

$$\alpha_0 \cdot \beta_{(j+1)t} - \beta_0 \cdot \alpha_{(j+1)t} \neq 0 \pmod{p-1},$$

$$g^{x_{(j+1)t}} \equiv x_j^{s_{(j+1)t}} \cdot y_{j,i_{j+1}}^{x_j} \quad (2')$$

After $x_k, y_{kt}, s_k, k = 1, 2, \dots, d, t = 1, 2$ are simulated, t_k, r_k are computed as follows:

$$g_1^{y_{k1}} g_2^{y_{k2}} \equiv x_k^{t_k} \cdot x_0^{r_k} \cdot g^{x_k} \pmod{p}$$

So,

$$g^{\gamma_1 \cdot y_{k1} + \gamma_2 \cdot y_{k2}} \cdot y^{\delta_1 \cdot y_{k1} + \delta_2 \cdot y_{k2}} \equiv g^{\alpha_k \cdot t_k + \alpha_0 \cdot r_k + x_k} \cdot y^{\beta_k \cdot t_k + \beta_0 \cdot r_k} \pmod{p}$$

Compute t_k, r_k satisfy the following system equations:

$$\begin{cases} \alpha_k \cdot t_k + \alpha_0 \cdot r_k + x_k \equiv \gamma_1 \cdot y_{k1} + \gamma_2 \cdot y_{k2} \pmod{p-1} \\ \beta_k \cdot t_k + \beta_0 \cdot r_k \equiv \delta_1 \cdot y_{k1} + \delta_2 \cdot y_{k2} \pmod{p-1} \end{cases}$$

From $\alpha_k \cdot \beta_0 - \alpha_0 \cdot \beta_k \neq 0 \pmod{p-1}$, the system equations has one solution. After x_d, s are selected, the simulator computes y_{d1} such that:

$$g^m \equiv x_d^s \cdot (y_{d1})^{x_d} \pmod{p}$$

The i -th simulated signature is:

$$S(m) = (Y_1, i_1, s_1, t_1, r_1; \dots, Y_d, i_d, s_d, t_d, r_d; m, s)$$

So, the simulator runs the whole simulating process by the above method. The simulating process is indistinguishable from a real signing process in polynomial time. Because in the real signature scheme, the signer selects $x_k, x_{k-1}, y_{k-1, i_k}$ which are indistinguishable from random integers in Z_p^* . s_k is the only solution of the equation (1'). In the simulated signature scheme, the simulator selects x_k, x_{k-1}, s_k which are indistinguishable from random integer in Z_p^* . y_{k-1, i_k} is the only solution of equation (2'). So, all the parameters $x_k, x_{k-1}, s_k, y_{k-1, i_k}$ in the simulated signature are indistinguishable from the corresponding parameters in the real signature. Because anyone besides the simulator don't know $y, \text{index}_{p,g}(g_1), \text{index}_{p,g}(g_2), \text{index}_{p,g}(x_k), \text{index}_{p,g}(x_0)$, he cannot distinguish t_j, r_j like the simulator, so t_j, r_j in the simulated signature are indistinguishable from that of the real signature. Therefore these two schemes are indistinguishable.

Theorem 2: If there is an algorithm that can forge a signature under adaptively chosen message attacks with a non-negligible probability, there is a probabilistic polynomial-time algorithm A to compute the discrete logarithm problem, i.e. given any random integer $y \in Z_p^*$, A can output $\text{index}_{p,g}(y)$ with a non-negligible probability.

Proof: Select any random integer $y \in Z_p^*$, We run the above entire simulated process. If an algorithm can forge a legitimate signature of the real signing scheme under adaptively chosen message attacks with an non-negligible probability, the algorithm can forge a legitimate signature of the simulated signature scheme with the same probability.

We provide that, after 2^d calls to the simulation, the attack algorithm forges a new signature different from the simulated signatures.

$$S(m') = (Y'_1, i'_1, s'_1, t'_1; \dots, Y'_d, i'_d, s'_d, t'_d; m', s')$$

For the same path, the simulated signature is:

$$S(m) = (Y_1, i_1, s_1, t_1; \dots, Y_d, i_d, s_d, t_d; m, s)$$

Provided that $S(m)$ and $S(m')$ have $(j - 1)$ internal nodes that have the same pairs (x_k, y_{k1}, y_{k2}) , $k = 1, 2$. Because the root is fixed, j must exist. For the j -node, $(x_j, y_{k1}, y_{k2}) \neq (x'_j, y'_{k1}, y'_{k2})$.

We prove the following 3 cases:

Case 1: If $x_j \neq x'_j$, We get the following two equation:

$$g^{x'_j} \equiv x_{j-1}^{s'_j} \cdot (y_{j-1,i_j})^{x_{j-1}} \pmod{p}$$

So,

$$g^{x'_j} \equiv g^{\alpha_{j-1} \cdot s'_j + u_{j-1,i_j} \cdot x_{j-1}} \cdot y^{\beta_{j-1} \cdot s'_j + v_{j-1,i_j} \cdot x_{j-1}} \pmod{p} \quad (3')$$

We get the following equation:

$$g^u \equiv y^v \pmod{p}$$

here

$$\begin{aligned} u &\equiv x'_j - \alpha_{j-1} \cdot s'_j - u_{j-1,i_j} \cdot x_{j-1} \pmod{p} \\ v &\equiv \beta_{j-1} \cdot s'_j + v_{j-1,i_j} \cdot x_{j-1} \pmod{p} \end{aligned}$$

Because $g.c.d(x_{j-1}, p - 1) = 1$, and the attacker don't know the parameters $\alpha_{j-1}, \beta_{j-1}, u_{j-1,i_j}, v_{j-1,i_j}$, so $g.c.d(v, q) = 1$ holds with the high probability of $1 - \frac{a}{p-1} = 1 - \frac{1}{q}$. We can provide that $g.c.d(v, q) = 1$.

If $g.c.d(v, p - 1) = 1$, we easily compute $index_{p,g}(y) \equiv u \cdot v^{-1} \pmod{p - 1}$.

If $g.c.d(v, p - 1) = t > 1$, t is small, we know $index_{p,g}(y)$ is a solution of the following equation:

$$u \equiv x \cdot v \pmod{p - 1} \quad (4')$$

So, we can compute the only solution x' of the following equation:

$$\frac{u}{t} \equiv x \cdot \frac{v}{t} \pmod{\frac{p-1}{t}}$$

So, equation $(4')$ has t solutions such that $x' + i \cdot \frac{p-1}{t}$, $i = 0, 1, \dots, t-1$. We can verify which solution is $index_{p,g}(y)$. Therefore $index_{p,g}(y)$ is easily computed.

Case 2: If $x_j = x'_j$, then $(y_{j1}, y_{j2}) \neq (y'_{j1}, y'_{j2})$. So,

$$g_1^{y_{j1}} \cdot g_2^{y_{j2}} \equiv x_j^{t_j} \cdot x_0^{r_j} \cdot g^{x_j} \pmod{p} \quad (5')$$

$$g_1^{y'_{j1}} \cdot g_2^{y'_{j2}} \equiv x_j^{t'_j} \cdot x_0^{r'_j} \cdot g^{x_j} \pmod{p} \quad (6')$$

From $(6')$, we get the following equation:

$$g^{\gamma_1 \cdot y'_{j1} + \gamma_2 \cdot y'_{j2}} \cdot y^{\delta_1 \cdot y'_{j1} + \delta_2 \cdot y'_{j2}} \equiv g^{\alpha_j \cdot t'_j + \alpha_0 \cdot r'_j + x_j} \cdot y^{\beta_j \cdot t'_j + \beta_0 \cdot r'_j} \pmod{p}$$

So, the following equation holds:

$$g^u \equiv y^v \pmod{p}$$

here

$$u \equiv \gamma_1.y'_{j1} + \gamma_2.y'_{j2} - \alpha_j.t'_j - \alpha_0.r'_j - x_j \pmod{p-1}$$

$$v \equiv -\delta_1.y'_{j1} - \delta_2.y'_{j2} + \beta_j.t'_j + \beta_0.r'_j \pmod{p-1}$$

Because $g.c.d(x_j, p-1) = 1$, and the attacker don't know $y, \alpha_t, \beta_t, t = 0, 1, 2; \gamma_t, \delta_t, t = 1, 2$. So, similar to case 1, we can provided that $g.c.d(u, q) = 1$, then $g.c.d(v, q) = 1$.

If $g.c.d(v, p-1) = 1$, we can easily compute $\text{index}_{p,g}(y) \equiv uv^{-1} \pmod{p-1}$.

If $g.c.d(v, p-1) = t$ is small, we compute $\text{index}_{p,g}(y)$ similar to Case 1.

Case 3: For $j = 1, 2, \dots, d$, $(x_j, y_{j,i_{j+1}}) = (x'_j, y'_{j,i_{j+1}})$. So, $m \neq m'$. The following equation holds:

$$g^{m-m'} \equiv x_d^{s-s'} \equiv (g^{\alpha_d}.y^{\beta_d})^{s-s'} \pmod{p}$$

Because $m, m' \in [0, q-1]$, $g.c.d(q, m-m') = 1$ holds. Similar to case 1, we can compute $\text{index}_{p,g}(y)$.

So, if there is an algorithm that can forge a signature under adaptively chosen message attack with a non-negligible probability, there is a probabilistic polynomial-time algorithm A to compute the discrete logarithm problem with a non-negligible probability. Thus the signature scheme is no existentially forgeable under adaptively chosen message attack.

4 Conclusion and Discussions

In this paper, we have introduced an alternate secure and efficient tree-structure signature scheme based on discrete logarithms, in addition to the scheme introduced in [4]. The scheme is secure and efficient. It is interesting to know that, although the original ELGamal signature scheme is existentially forgeable, it can be used to construct secure tree-structure signature scheme as the RSA does. (A more detail analysis of the differences as well as the advantages/disadvantages between our scheme and that in [4] will be presented in the conference.)

Designing a secure and efficient tree-based signature scheme has the additional advantage that although the difficulty of both the discrete logarithm and RSA problems are yet unproved, discrete logarithm problem seems to be "harder". One reason is that solving the discrete logarithm of Z_n^* implies solving the factorization problem in Z_n^* .

References

1. G. B. Agnew, R. C. Mullin, S. A. Vanstone, Improved digital signature scheme based on discrete exponentiation, Electronics Letters, Vol. 26, pp. 1024-1025, 1990. 167
2. M. Bellare, P. Rogaway, The Exact Security of Digital Signatures-How to Sign with RSA and Rabin, Advances in Cryptology-EUROCRYPT'96, pp.399-417. 167
3. J. Bos and D. Chaum, Provably Unforgeable Signatures, Pro. Advances in Cryptology-Crypto'92 proceedings, Springer Verlag, 1993, pp. 1-14.

4. R.Cramer, I. Damgard: Secure Signature Schemes based on Interactive Protocols, Proceedings of Crypto'95, Springer Verlag, pp.329-342. **167, 168, 176**
5. R. Cramer and Ivan Damgard, New Generation of Secure and Practical RSA-based Signatures, Crypto'96,pp.173-185. **167, 168, 172**
6. C. Dwork, Moni Naor, An Efficient Existentially Unforgeable Signature Scheme and Its Applications, Crypto'94, pp. 234-246. **167, 168, 172**
7. T. ElGamal, A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithm, IEEE Transactions on Information Theory, IT-31(4):469-472, 1985. **167, 168**
8. R. Gennaro, S. Halevi, T. Rabin, Secure Hash-and-Sign Signatures Without the Random Oracle, Advances in Cryptology-EUROCRYPT'99, pp.123-139.
9. R. Gennaro, H. Krawczyk, T. Rabin, RSA-based Undeniable Signature, Pro. Advances in Cryptology-Crypto'97, Springer Verlag, 1998, pp.133-148.
10. O. Goldreich, Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme, Proceedings of Crypto'86, pp.104-110. **172**
11. S. Goldwasser, S. Micali and R. Rivest ,A Digital Signature Scheme Secure against Chosen Message Attacks, SIAM Journal on Computing, 17(2):281-308, 1988. **167, 168, 169, 172**
12. P. Horster, H. Petersen, M. Michels, Meta-ELGamal signature schemes, 2nd ACM Conference on Computer and Communications Security, pp.96-107, 1994.
13. R. C. Merkle, A Certified Digital Signature, Proceedings of Crypto'89, Springer Verlag LNCS series, pp.234-246. **168**
14. M. Michels and M. Stadler, Generic Constructions for Secure and Efficient Confirmer Signature Schemes, Advances in Cryptology- EUROCRIPT'98, pp.406-421. **167**
15. National Institute of Standards and Technology, federal Information Process Standard, FIPS PUB 186-1: Digital Signature Standard (DSS), 1991. **167**
16. K. Nyberg, R. Rueppel, message recovery for signature schemes based on the discrete logarithm problem, Eurocrypto'94, pp.182-193, 1994. **167**
17. K.Ohta, T. Okamoto, On Concrete Security Treatment of Signatures Derived from Identification, Advances on Cryptology-CRYPTO'98, pp. 354-369.
18. T. Okamoto: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes, Proceedings of Crypto'92, Springer Verlag, pp.31-53.
19. D. Pointcheval, J.Stern, Secure Proofs for Signature Schemes, Advances in Cryptology-eurocrypt,96, pp.387-399. **167**
20. R. Rivest, A. Shamir, L. Adleman, A Method for Obtaining Digital Signatures and Public Key Cryptosystem, Communications of ACM, 21(1978), pp. 120-126. **167, 168**
21. S.M. Yen, C. S. Laih, New Digital Signature Scheme based on Discrete Logarithm, Electronics Letters, Vol. 29, No. 12, pp.1120-1121, 1993. **167**

All-or-Nothing Transform and Remotely Keyed Encryption Protocols

Sang Uk Shin, Weon Shin, and Kyung Hyune Rhee

Department of Computer Science, Pukyong National University,
599-1, Daeyeon-dong, Nam-gu, Pusan, 608-737, Korea
`{shnsu,redcomet,khrhee}@woongbi.pknu.ac.kr`
`http://unicorn.pknu.ac.kr/~{soshin, redcomet and khrhee}`

Abstract. We propose two new Remotely Keyed Encryption protocols; a length-increasing RKE and a length-preserving RKE. The proposed protocols have smaller computations than the existing schemes and provide sufficient security. In proposed protocols, we use the cryptographic transform with all-or-nothing properties proposed by Rivest as encryption operation in the host. By using all-or-nothing transform(AONT), RKE can be more dependent on the entire input and all encryption operation can be implemented only by using hash functions in the host. Moreover, we provide a length-preserving RKE protocol requires only one interaction between the host and the card.

1 Introduction

No cryptographic protocol is stronger than the mechanism protecting its secret keys. However, many computer and communication systems do not have a safe place which can store secret keys and perform the cryptographic computations. Especially, computers connected to the network such as Internet may be partly controlled by an adversary. Therefore, it is nature to consider adding an external, special-purpose device such as a smart card. Since such devices have one purpose and communicate via a limited set of functions, they can be made much more secure than general-purpose host machines. However, it is not practical to rely on such devices to perform all sensitive cryptographic operations. Such devices have limited bandwidth, memory, and processor speed.

A Remotely Keyed Encryption(RKE) focuses the fact that many high-bandwidth applications need symmetric-key encryption schemes that store long-lived keys in low-bandwidth smart-cards. A RKE can do bulk encryption/decryption for high-bandwidth applications in a way that takes advantage of both the superior power of the host and the superior security of the smart-card. A RKE consists of two protocols, one for encryption and one for decryption. Given an l -bit input, either to encrypt or to decrypt, a protocol runs; the host sends a challenge value to the card depending on the input, and the card replies a response value depending on the challenge value and the key. During the run of a protocol, every challenge value may depend on the input and the previous response values, and the response values may depend on the key and the previous challenge values.

In this paper, we propose two new remotely keyed encryption protocols; a length-increasing RKE and a length-preserving RKE. The proposed protocols have smaller computations than the existing schemes and provide sufficient security. And in proposed protocols, we use the cryptographic transform with all-or-nothing properties proposed by Rivest[10] as encryption operation in the host. As All-Or-Nothing Transform(AONT) with all-or-nothing properties, we use an improved version of the AON hashing-3 proposed by Shin, et al[11]. The improved AON hashing-3(IAON hashing) provides better security without additional computation overhead. By using the AONT, RKE can be more dependent on the entire input and all encryption operation can be implemented only by using hash functions in the host. Also, we suggest a solution for an open problem suggested by Blaze, Feigenbaum and Naor[4][5]:

Is there a secure, length-preserving RKE that requires only one round of interaction?

The proposed length-preserving RKE requires only one interaction between the host and the card.

The remainder of this paper is organized as follows. In section 2, we describe AONT and propose an improved AON hashing. In section 3, we describe the definition and security requirements of RKE. And in section 4, we propose two new secure RKE protocols, a secure length-increasing RKE and a secure length-preserving RKE. Finally, we have conclusions in section 5.

2 All-or-Nothing Transform

In 1997, Rivest proposed an all-or-nothing encryption, a new encryption mode for block ciphers[10]. This mode has the property that one must decrypt the entire ciphertext before one can determine even one message block. This means that brute-force searches against all-or-nothing encryption are slowed down by a factor equal to the number of blocks in the ciphertext.

Rivest proposed the all-or-nothing transform which is referred to "package transform", as follows[10]:

- (1) Let the input message be m_1, m_2, \dots, m_s .
- (2) Choose at random a key K for the package transform block cipher.
- (3) Compute the output sequence $m'_1, m'_2, \dots, m'_{s'}$ for $s' = s + 1$ as follows:
 - $m'_i = m_i \oplus E(K, i)$ for $i = 1, 2, 3, \dots, s$.
 - Let

$$m'_{s'} = K \oplus h_1 \oplus \dots \oplus h_s,$$

where

$$h_i = E(K_0, m'_i \oplus i) \text{ for } i = 1, 2, \dots, s,$$

where K_0 is a fixed, publicly-known encryption key.

The block cipher for the package transform does not use a secret key, and needs not be the same as the block cipher for encrypting the pseudo-message. We assume that the key space for the package transform block cipher is sufficiently large that brute-force searching for a key is infeasible. It is easy to see that the package transform is invertible:

$$K = m'_{s'} \oplus h_1 \oplus \dots \oplus h_s,$$

$$m_i = m'_i \oplus E(K, i) \text{ for } i = 1, 2, \dots, s.$$

If any block of pseudo-message sequence is unknown, the K can not be computed, and so it is infeasible to compute any message block.

An all-or-nothing transform is merely a pre-processing step, and so it can be used with already-existing encryption software and device, without changing the encryption algorithm. The legitimate communicants pay a penalty of approximately a factor of three in the time it takes them to encrypt or decrypt in all-or-nothing mode, compared to an ordinary separable encryption mode. However, an adversary attempting a brute-force attack pays a penalty of a factor of t , where t is the number of blocks in the ciphertext.

As an application of AONT, Shin, et al proposed hash functions and its application to the MAC(Message Authentication Code) with all-or-nothing properties in 1999[11]. The proposed AON hashings used the existing hash functions without changing their algorithm and were secure against the existing known attacks. Also they can be easily applied to the MAC which can provide both the authentication and the confidentiality for message by using only hash functions. Now we describe shortly the proposed AON hashing-3 which operates as following. For details of the algorithm, see [11]:

A. Sender

- (1) Splitting an input message X into s blocks of n -bit each : $Y = (X_1, X_2, \dots, X_s)$.
- (2) Generating a random key K of n -bit.
- (3) Compute a pseudo-message Y by all-or-nothing transform.

$$Y_0 = IV, X_0 = 0,$$

$$Y_i = X_i \oplus h_{Y_{i-1}}(K \oplus (\overline{X_{i-1}||i})) , \quad i = 1, 2, \dots, s.$$

- (4) Compute the last pseudo-message block, Y_{s+1} (n -bit length).

$$MD = h_{K_p}(Y_1 || \dots || Y_s || h_{IV}(K_p)) ,$$

$$Y_{s+1} = K \oplus \{\overline{MD}\}.$$

- (5) Send $(Y || h_{MD}(Y_{s+1}))$.

B. Receiver

- (1) Receive $(Y || Z)$.

- (2) Splitting the pseudo-message Y into $s+1$ blocks of n -bit each, Y_1, Y_2, \dots, Y_s and Y_{s+1} .
- (3) Recover the random key K .

$$MD' = h_{K_p}(Y_1 \parallel \dots \parallel Y_s \parallel h_{IV}(K_p)),$$

$$K = Y_{s+1} \oplus \{\overline{MD'}\}.$$

- (4) Check if $Z = h_{MD'}(Y_{s+1})$.
- (5) Recover the original message.

$$Y_0 = IV, X_0 = 0,$$

$$X_i = Y_i \oplus h_{Y_{i-1}}(K \oplus (\overline{X_{i-1} \parallel i})), i = 1, 2, \dots, s.$$

We propose an improved AON hashing (IAON hashing) which provides better security without additional computation overhead. IAON hashing operates as following:

A. Sender

- (1) Compute $K = h_{IV}(X)$.
- (2) Splitting an input message X into s blocks of n -bit each : $Y = (X_1, X_2, \dots, X_s)$.
- (3) Compute a pseudo-message Y by all-or-nothing transform.

$$Y_0 = IV, X_0 = K,$$

$$Y_i = X_i \oplus h_{X_{i-1}}(Y_{i-1} \parallel (K \oplus i)), i = 1, 2, \dots, s.$$

- (4) Compute the last pseudo-message block, Y_{s+1} (n -bit length).

$$MD = h_{K_p}(Y_1 \parallel \dots \parallel Y_s \parallel h_{IV}(K_p \oplus (s+1))),$$

$$Y_{s+1} = K \oplus MD.$$

- (5) Send $(Y \parallel h_{MD}(Y_{s+1}))$.

B. Receiver

- (1) Receive $(Y \parallel Z)$.
- (2) Splitting the pseudo-message Y into $s+1$ blocks of n -bit each, Y_1, Y_2, \dots, Y_{s+1} .
- (3) Recover K .

$$MD' = h_{K_p}(Y_1 \parallel \dots \parallel Y_s \parallel h_{IV}(K_p \oplus (s+1))),$$

$$K = Y_{s+1} \oplus MD'.$$

- (4) Check if $Z = h_{MD'}(Y_{s+1})$.
- (5) Recover the original message.

$$Y_0 = IV, X_0 = K,$$

$$X_i = Y_i \oplus h_{X_{i-1}}(Y_{i-1} \parallel (K \oplus i)), i = 1, 2, \dots, s.$$

(6) Check if $K = h_{IV}(X_1 \dots X_s)$.

In IAON hashing, each pseudo-message block depends on the entire original message since K is a hash value of the entire original message, and it is generated as k -bit random number in AON hashing-3. Thus, IAON hashing does not require an additional computation.

To find a collision pair, in AON hashing-3, an adversary has to find the collision pair of the pseudo-message and then search the random key K mapping the input message pair to collision pseudo-message. However, IAON hashing imposes stronger restriction to find a collision pair since K is a hash value of the original input message. That is, an attacker has to find a collision pair of the pseudo-message and then check whether the modified input message is mapped to the collision pair of the pseudo-message using the hash value of the modified input message. At the worst case, AON hashing-3 requires $2^{n/2+k}$ operations but IAON hashing requires $2^{n/2+\beta}$ operations to find a collision pair (β is a length of the input message).

IAON hashing can be easily applied to the MAC (Message Authentication Code). In this case, it is possible to provide both the authentication and confidentiality for message. For this MAC application, two communication parties between sender and receiver have to securely keep publicly-known random constant K_p . This MAC construction may be considered as the variant of HMAC proposed by Bellare, et al [1] such like that

$$HMAC_k(x) = h(\bar{k} \oplus opad, h(\bar{k} \oplus ipad, x))$$

where k is a secret information which held by two parties.

Thus AON-MAC has the same security as that of HMAC. Furthermore, an attacker who does not know a random constant K_p cannot find the K that is needed to recover the entire original message. To find a K , an attacker has to find the K_p such that

$$MD = h_{K_p}(Y_1 || \dots || Y_s || h_{IV}(K_p \oplus (s+1))) ,$$

The best known attack for this scheme is the divide-and-conquer attack proposed by Preneel and van Oorschot [9]. Also, to decrypt only one block of message, an attacker who does not know the K has to solve the following:

$$X_i = Y_i \oplus h_{X_{i-1}}(Y_{i-1} || (K \oplus i))$$

To solve it, he must guess $h_{X_{i-1}}(Y_{i-1} || (K \oplus i))$ or find K and X_{i-1} .

We can improve the security by adding some overhead to the above schemes. We can encrypt the last pseudo-message block and message digest pair $(Y_{s+1}, h_{MD}(Y_{s+1}))$ using the block cipher with the secret key K_S , and send it to the recipient. This scheme can provide the confidentiality of message by encrypting only some blocks without encrypting the entire message. It is specially useful if the cipher is a public key cryptosystem, such as RSA or ElGamal. In this scheme, we can use RSA to securely encrypt message longer than the key size without

need for a symmetric cipher. An additional encryption overhead is $2n$ bits. If we use SHA-1 as a building block, the length of the encrypting block becomes a size of total of 360 bits. Therefore, the performance degradation can be negligible in this case. However an attacker must find the K_S and K_p for decrypting the entire message.

IAON-MAC is similar to the concept of "Scramble All, Encrypt Small" proposed by Jakobsson, et al[6].

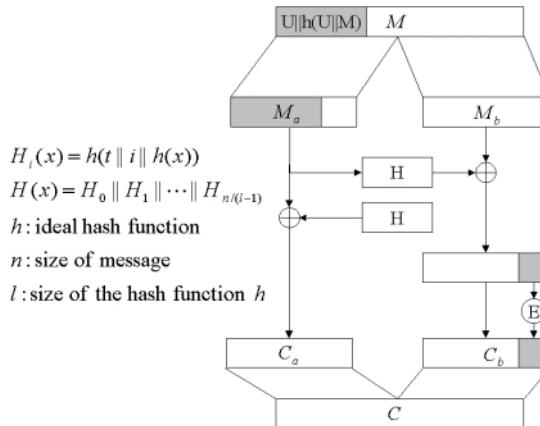


Fig. 1. Scramble All, Encrypt Small

In IAON-MAC, the process that transforms the original message into the pseudo-message is considered as the scrambling process in [6]. We compare both schemes in view point of computational amounts. Assuming a 160,000 bits message is processed, the Jakobsson's scheme requires 1001 computations of a hash function. That is, about 313,313 computations of a compression function of a hash function. However, IAON-MAC requires about 1,627 computations of a compression function. Thus, IAON-MAC has less computations than the Jakobsson's scheme.

3 Remotely Keyed Encryption Protocols

A remotely keyed encryption was first proposed by Blaze[3]. Blaze focuses the fact that many high-bandwidth applications need symmetric-key encryption schemes that store long-lived keys in low-bandwidth smart-cards. A RKE can do bulk encryption/decryption for high-bandwidth applications in a way that takes advantage of both the superior power of the host and the superior security of the smart-card. A RKE consists of two protocols, one for encryption and one for decryption. Given an l -bit input, either to encrypt or to decrypt, a protocol runs: the host sends a challenge value to the card depending on the input, and

the card replies a response value depending on the challenge value and the key. During the run of a protocol, every challenge value may depend on the input and the previous response values, and the response values may depend on the key and the previous challenge values.

3.1 Related Work

Blaze was the first to use the term "remotely keyed encryption" and proposed a specific scheme[3]. It is based on the idea of letting the host send the card one block which depends on the whole message. This block would be encrypted with the smart card secret key, and would also serve as a seed for the creation of a temporary secret key that will be used by the host in order to encrypt the rest of the message. However, in 1997 Lucks pointed that Blaze's scheme had problems in that allowed an adversary to forge a new valid plaintext/ciphertext pair after several interactions with the card[7]. Lucks suggested an alternative method[7] but was attacked by Blaze, Feigenbaum and Naor who further demonstrated the subtleties of this problem[4][5]. They showed that the encryption key used for the largest part of the message is deterministically derived from the two first blocks of the message, hence an adversary who takes control of the host will be able to decrypt a large set of messages with only one query. They derived a careful formal model and a scheme based on pseudorandom functions. In 1999, Lucks further extended their security model and proposed a faster one[8]. And Jakobsson, et al proposed "Scramble All, Encrypt Small" notion[6]. It scrambles the message in a publicly available method and then deprives an adversary of the ability to invert the scrambling by letting the smart card encrypt just a small part of the scrambled message.

3.2 Security Requirements

Lucks proposed that a RKE should have three properties[8]:

- (i) Forgery security : If the adversary has controlled the host for m interactions, then it cannot produce $m + 1$ plaintext/ciphertext pairs.
- (ii) Inversion security : Access to encryption should not imply the ability to decrypt and vice versa.
- (iii) Pseudorandomness : The encryption function should behave randomly, for someone neither having access to the card, nor knowing the secret key.

Requirements (i) and (ii) restrict the abilities of an adversary with access to the smart card. Requirement (iii) is only valid for outside adversaries, having no access to the card. If an adversary could compute forgeries or run inversion attacks, she could easily distinguish the encryption function from a random one.

Recently, Blaze, Feigenbaum and Naor found a better formalism to define the pseudorandomness of RKEs[4][5]. Their idea is based on the adversary gaining direct access to the card for a certain amount of time, making m interaction with the card. For the adversary having lost direct access to the card, the encryption function should behave randomly. An attacker is divided into two phases:

- (i) The *host phase* : During the host phase, an attacker A may have a total of m interactions with the card. He may send any message to the card during one of these interaction and may deviate from the protocol. However, since m is an upper bound on the total number of interactions, A obtains at most m plaintexts and m ciphertexts during the host phase. After these m interactions with the card, A loses control of the host.
- (ii) The *distinguishing phase* : An attacker chooses plaintexts or ciphertexts and asks for their encryptions or decryptions. The answer to these queries are either chosen randomly, or by honestly encrypting or decrypting according to the RKE. An adversary's task is to distinguish between the random case and honest encryption.

If, during the distinguishing phase, an adversary A queries the oracle about any of the m plaintexts and m ciphertexts that he obtained during the host phase, her task would be quite easy. Thus, in the distinguishing phase, it needs to filter texts that appeared in the host phase before. To do this, BFN introduced an "arbiter" algorithm B . The purpose of the arbiter B is to make sure that A does not ask during the distinguishing phase any of the queries that it asked during the host phase.

According to Lucks and BFN, the formal definitions of a RKE can be given as follows[4][5][8].

Definition 1 (length-preserving RKE). A length-preserving RKE is a pair of protocols, one for encryption and one for decryption, to be executed by a host and a card. The length of a ciphertext must be the same as that of the corresponding plaintext. Let B be an algorithm, the "arbiter algorithm", which is initialized with a transcript of the communication between the host and the card during the host phase. During the host phase, an attacker A may play the role of the host in a total of q_h interactions with the card. During this phase, A may send any message to the card and does not necessarily follow the encryption or decryption protocol. During the distinguishing phase, A choose up to q_d texts T as queries and asks for the corresponding encryptions or decryptions. W.l.o.g., we prohibit A to ask equivalent queries, i.e., to ask twice for the encryption of T , to ask twice for the decryption of T , or ask once for the encryption of a T and some time before or after this for the decryption of the corresponding ciphertext. Before the distinguishing phase starts, a switch S is randomly set either to 0 or to 1. If the arbiter B acts, A 's query s answered according to the RKE. If B does not act on, the answers are generated depending on S . Consider A asking for the encryption or decryption of a text $T \in \{0, 1\}^\beta$ with $\beta > b$. If $S = 0$, the response is evaluated according to the RKE. If $S = 1$, the response is a random value in $\{0, 1\}^\beta$. After the distinguishing phase, A 's task is to guess S . The difference between the probability that A accepts on a continuation of the RKE and the probability that A accepts on a switch to a random function pair must be negligible.

Length-increasing RKEs would be easier to construct than length-preserving RKEs. However, we can require additional security properties of length-in-

creasing RKEs that are not achievable in the length-preserving case. In the length-increasing case, each plaintext may correspond to multiple ciphertexts because the ciphertext space is bigger than the plaintext space. For a length-increasing RKE, BFN introduced the concept of a "random, self-validating black box" [5]. Given that an arbiter is filtering based on a transcript of the host phase communication, after the host phase, the adversary cannot tell whether he is interacting with real protocols or with a "random, self-validating black box." A "random, self-validating black box" contains an encryption box and a decryption box. On any input of the appropriate plaintext length, the encryption box outputs a random string of the appropriate ciphertext length. The decryption box outputs "invalid" on all inputs, except those that were previously output by the encryption box, and on those it outputs the input string on which the encryption box gave this output.

Definition 2 (length-increasing RKE). *A length-increasing RKE is a pair of protocols, one for encryption and one for decryption, to be executed by a host and a card. The length of a ciphertext is greater than the length of the corresponding plaintext. If its input is a ciphertext that has previously been output by the encryption protocol, the decryption protocol outputs the corresponding plaintext; otherwise, it may output "invalid". The RKE is secure if there is a polynomial-time arbiter B that can enforce the following restriction on any adversary A: During the host phase, an attacker A may play the role of the host in a total of q_h interactions with the card. During this phase, A may send any message to the card and does not necessarily follow the encryption or decryption protocol. Between the host phase and the distinguishing phase, a choice is made between continuing to use the RKE or switching to a random, self-validating black box. B gets as input the transcript of the host phase communication between the host and the card. During the distinguishing phase, A choose up to q_d texts T as queries and asks for the corresponding encryptions or decryptions. Before the distinguishing phase starts, a switch S is randomly set either to 0 or to 1. If the arbiter B acts, A's query s answered according to the RKE. If B does not act on, the answers are generated depending on S. Consider A asking for the encryption or decryption of a text $T \in \{0, 1\}^\beta$ with $\beta > b$. If $S = 0$, the response is evaluated according to the RKE. If $S = 1$, the response is evaluated according to a random, self-validating black box. The difference between the probability that A accepts on a continuation of the RKE and the probability that A accepts on a switch to a random, self-validating black box must be negligible.*

4 New Secure Remotely Keyed Encryption Schemes

4.1 Building blocks

In this section, we describe the building blocks that we use for new secure remotely keyed encryption(SRKE) schemes. As will be proven below, our schemes are secure if their building blocks are secure. The following is a description of the building blocks.

- X and Y denote the plaintext and the ciphertext, respectively. Usually, both are given in blocks and hence are denoted $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_n)$ where each of X_i and Y_i is in $\{0, 1\}^b$.
- E and D denote the encryption and decryption functions of a block cipher. $E_k(X_i)$ denotes the encryption of plaintext block X_i with encryption key k , i.e., $E_k, D_k : \{0, 1\}^b \rightarrow \{0, 1\}^b$ and $D_k(E_k(X_i)) = X_i$. The required security property is that it should be a strong pseudorandom permutation. For $k \neq k'$, the permutations E_k and $E_{k'}$ are independent.
- $F : \{0, 1\}^b \rightarrow \{0, 1\}^b$ is a pseudorandom function. It may or may not be identical to the encryption function E of the block cipher. In situation that never require the function to be inverted, we use F_s rather than E_s . Similarly, two random functions F_k and $F_{k'}$ depending on independently chosen keys k and k' are independent.
- G_K denotes the encryption of an n -block plaintext (X_1, \dots, X_n) using encryption key K . The corresponding decryption function is denoted \hat{G}_K . The security requirement for G_K is that if K is chosen uniformly at random, then $G_K(X_1, \dots, X_n)$ is pseudorandom for any (X_1, \dots, X_n) . A possible example of G_K is to apply a pseudorandom generator to K and Xor the resulting sequence with (X_1, \dots, X_n) . Another example is to use E_K with some sort of chaining, e.g., CBC[2].
- $h : \{0, 1\}^* \rightarrow \{0, 1\}^b$ is a collision-resistant hash function.

For the analysis, we assume the used building blocks(such as block ciphers) to behave like their ideal counterparts(such as random permutations).

4.2 A Secure, Length-Increasing RKE

Using the above building blocks, we first propose a secure, length-increasing RKE(LI_SRKE) protocol. The private key stored in the smart card is denoted k_1 , k_2 and k_3 . The encryption protocol works as follows:

LI_SRKE encryption protocol : input $X = (X_1, \dots, X_l)$; output $t, Y_0, Y = (Y_1, \dots, Y_l)$

- (1) Host : $md_i \leftarrow h(X_1, \dots, X_l)$, $K_0 \leftarrow md_i$
- (2) Host : $Y \leftarrow G_{K_0}(X)$
- (3) Host : $md_o \leftarrow h(Y)$
- (4) Host \rightarrow Card : K_0, md_o
- (5) Card : $Y_0 \leftarrow E_{k_1}(K_0 \oplus F_{k_2}(md_o))$
- (6) Card : $t \leftarrow F_{k_2}(md_o) \oplus F_{k_3}(K_0)$
- (7) Card \rightarrow Host : Y_0, t

The decryption protocol works as follows:

LI_SRKE decryption protocol : input $t, Y_0, Y = (Y_1, \dots, Y_l)$; output $X = (X_1, \dots, X_l)$ or "invalid"

- (1) Host : $md_o \leftarrow h(Y)$
- (2) Host \rightarrow Card : t, Y_0, md_o

- (3) Card : $K_0 \leftarrow D_{k_1}(Y_0) \oplus F_{k_2}(md_o)$
- (4) Card : if $t \neq F_{k_2}(md_o) \oplus F_{k_3}(K_0)$ then $K_0 \leftarrow$ "invalid"
- (5) Card \rightarrow Host : K_0
- (6) Host : If $K_0 =$ "invalid" then output "invalid"
else $X \leftarrow \hat{G}_{K_0}(Y)$ if $K_0 = h(X)$ then output X

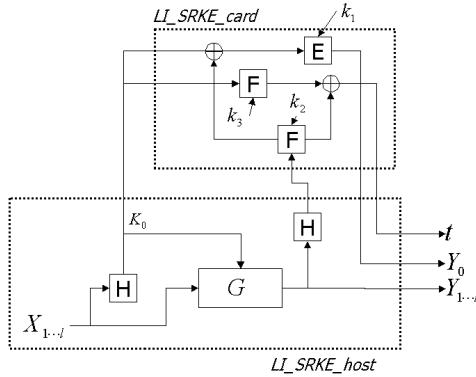


Fig. 2. LI_SRKE encryption protocol

As required by Definition 4, the arbiter B does not filter queries of (X_1, \dots, X_l) during the distinguishing phase. If no switch was made between phases it just sends them to the encryption protocol and to a random, self-validating black box if a switch was made. On the queries of (Y_1, \dots, Y_l) , B computes $h(Y_1, \dots, Y_l)$ and checks whether $(h(Y_1, \dots, Y_l), Y_0, t)$ occurs in the transcript of the host phase. If it does, then B sends the query to the decryption protocol, regardless of whether a switch was made between phases. If it doesn't, then B sends it either to the decryption protocol or to the random, self-validating black box, depending on whether a switch was made.

We analyze the security of LI_SRKE based on the analysis of BFN's length-increasing RKE scheme[5] as following:

Theorem 1. *LI_SRKE is a secure, length-increasing RKE protocol.*

Proof. The definition of the cryptographic building blocks E , F and G imply that any sequence of encryptions is indistinguishable from a random one. Consider the case of decryption queries. Suppose that a query (Y_1, \dots, Y_l) does not correspond to an encryption query (X_1, \dots, X_l) that occurred earlier in the distinguishing phase and $(md_o = h(Y_1, \dots, Y_l), Y_0, t)$ did not appear in the host phase. A random, self-validating black box will answer such a query by saying "invalid." The real protocol will also answer "invalid" if $t \neq F_{k_2}(md_o) \oplus F_{k_3}(K_0)$ ($K_0 = D_{k_1}(Y_0) \oplus F_{k_2}(md_o)$), but it will produce a decryption if $t = F_{k_2}(md_o) \oplus F_{k_3}(K_0)$.

Thus an attacker can tell whether a switch was made between phases only if it can find $(t, Y_0, Y = (Y_1, \dots, Y_l))$ such that

$$t = F_{k_2}(md_o) \oplus F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(md_o))$$

The collision resistance of h implies that the adversary cannot find $(Y_1, \dots, Y_l) \neq (Y'_1, \dots, Y'_l)$ such that $h(Y_1, \dots, Y_l) = h(Y'_1, \dots, Y'_l)$. Therefore, an attacker has to find collision pairs $(Y_0, h(Y_1, \dots, Y_l)) \neq (Y'_0, h(Y'_1, \dots, Y'_l))$ such that

$$\begin{aligned} & F_{k_2}(h(Y_1, \dots, Y_l)) \oplus F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(h(Y_1, \dots, Y_l))) \\ &= F_{k_2}(h(Y'_1, \dots, Y'_l)) \oplus F_{k_3}(D_{k_1}(Y'_0) \oplus F_{k_2}(h(Y'_1, \dots, Y'_l))) \end{aligned}$$

Suppose that F_{k_2} and F_{k_3} are truly random functions. Then all the values $F_{k_2}(h(Y_1, \dots, Y_l))$ and $F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(h(Y_1, \dots, Y_l)))$ are random values. Thus, the probability that $(Y_0, h(Y_1, \dots, Y_l)) \neq (Y'_0, h(Y'_1, \dots, Y'_l))$ but

$$\begin{aligned} & F_{k_2}(h(Y_1, \dots, Y_l)) \oplus F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(h(Y_1, \dots, Y_l))) \\ &= F_{k_2}(h(Y'_1, \dots, Y'_l)) \oplus F_{k_3}(D_{k_1}(Y'_0) \oplus F_{k_2}(h(Y'_1, \dots, Y'_l))) \end{aligned}$$

is $1/2^b$. There are $(q_h + q_d)^2$ possible pairs. Therefore, with the probability of $\frac{(q_h + q_d)^2}{2^b}$, the adversary can distinguish between a random, self-validating black box and the original encryption algorithm. ♠

The implementation of the function G

A function G which is used in the host of LLSRKE algorithm has to satisfy the pseudorandomness. Possible realization of G are a pseudorandom generator and a block cipher with some sort of chaining. As an alternative scheme, we propose a IAON_G function which is a modified version of IAON hashing described in section 2. IAON_G works as follows:

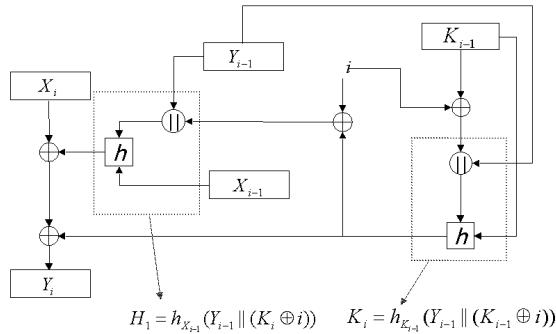
IAON_G encryption : input $X = (X_1, \dots, X_l)$; output $Y = (Y_1, \dots, Y_l)$

$$\begin{aligned} K_0 &= h(X) \\ X_0 &= K_0, Y_0 = h(K_0) \\ \text{for } i &= 1 \dots l \\ K_i &= h_{K_{i-1}}(Y_{i-1} || (K_{i-1} \oplus i)) \\ H_1 &= h_{X_{i-1}}(Y_{i-1} || (K_i \oplus i)) \\ Y_i &= X_i \oplus H_1 \oplus K_i \end{aligned}$$

IAON_G decryption : input $K_0, Y = (Y_1, \dots, Y_l)$; output $X = (X_1, \dots, X_l)$

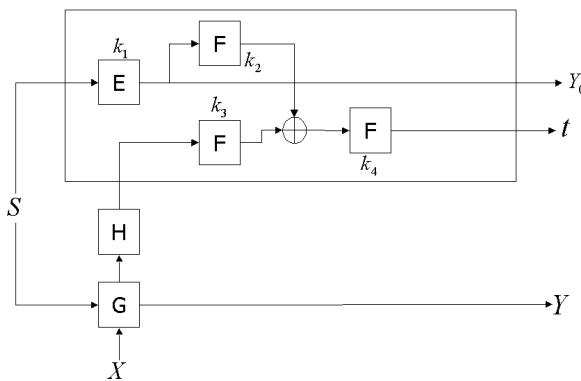
$$\begin{aligned} X_0 &= K_0, Y_0 = h(K_0) \\ \text{for } i &= 1 \dots l \\ K_i &= h_{K_{i-1}}(Y_{i-1} || (K_{i-1} \oplus i)) \\ H_1 &= h_{X_{i-1}}(Y_{i-1} || (K_i \oplus i)) \\ X_i &= Y_i \oplus H_1 \oplus K_i \end{aligned}$$

IAON_G is a modified version of AONT using hash functions, which transforms the original input message into the pseudo-message. An attacker who does

**Fig. 3.** IAON_G encryption process

not know K_0 cannot compute K_i and H_1 . Thus he cannot recover X_i . This function is considered as a pseduorandom generator using hash functions. Since hash functions have pseudorandomness, K_0 is considered as pseudorandom value, and H_1 and K_i are also pseudorandom values, respectively. Therefore IAON_G has pseudorandomness stated by BFN. Also by using IAON_G, the output of the proposed RKE becomes more dependent on the entire input message and all encryption operation on the host can be implemented only by using the hash function.

We compare the proposed LI_SRKKE with BFN's length-increasing RKE[5]. BFN's length-increasing RKE operates as Fig. 4.

**Fig. 4.** BFN's length-increasing RKE

On the part of the host, the proposed scheme requires one more computation of a hash function but it is considered as the computation of a random number S

in BFN's scheme. The communication between the host and the card is exactly the same for both protocols. Inside the card, LLSRKE needs three evaluations of cryptographic functions(such as E and F). In contrast to this, BFN's scheme needs four evaluations of cryptographic functions. In the case of BFN's scheme, to decrypt the ciphertext without interactions with the card, an attacker has to compute $S = D_{k_1}(Y_0)$. However, LLSRKE requires the computation of $K_0 = D_{k_1}(Y_0) \oplus F_{k_2}(md_o)$. Moreover, since LLSRKE uses IAON-G with all-or-nothing properties, LLSRKE is more dependent on the input message and more secure against key search attacks such as brute-force attack.

4.3 A Secure, Length-Preserving RKE

In this section, we propose a secure, length-preserving RKE(LP_SRKE) protocol. The private key stored in the smart card is denoted k_1, k_2 and k_3 . The encryption protocol works as follows:

LP_SRKE encryption protocol : input $X = (X_1, \dots, X_l)$; output $Y = (Y_1, \dots, Y_l)$

- (1) Host : $K_0 \leftarrow X_1 \oplus h(X_2, \dots, X_l)$
- (2) Host : $(Y_2, \dots, Y_l) \leftarrow G_{K_0}(X_2, \dots, X_l)$
- (3) Host : $t \leftarrow h(Y_2, \dots, Y_l)$
- (4) Host → Card : K_0, t
- (5) Card : $Y_1 \leftarrow E_{k_1}(K_0 \oplus F_{k_3}(t)) \oplus F_{k_2}(t)$
- (6) Card → Host : Y_1

The decryption protocol works as follows:

LP_SRKE decryption protocol: input $Y = (Y_1, \dots, Y_l)$; output $X = (X_1, \dots, X_l)$

- (1) Host : $t \leftarrow h(Y_2, \dots, Y_l)$
- (2) Host → Card : Y_1, t
- (3) Card : $K_0 \leftarrow D_{k_1}(Y_1 \oplus F_{k_2}(t)) \oplus F_{k_3}(t)$
- (4) Card → Host : K_0
- (5) Host : $(X_2, \dots, X_l) \leftarrow \hat{G}_{K_0}(Y_2, \dots, Y_l)$
- (6) Host : $X_1 \leftarrow K_0 \oplus h(X_2, \dots, X_l)$

We uses the IAON_G function as the LLSRKE protocol. We analyze the security of LP_SRKE based on the security analysis of BFN's length-preserving RKE[4][5] and Lucks's ARKE(Accelerated RKE)[8]. By X^i, Y^i, K_0^i, t^i , we denote the challenge and response values of the i -th protocol execution. The protocol can either be executed in the host phase indicated by $i \in \{1, \dots, q_h\}$, or in the distinguishing phase indicated by $i \in \{q_h + 1, \dots, q\}$.

Theorem 2. *LP_SRKE is a secure, length-preserving RKE protocol.*

Proof. For the proof, we first define the arbiter algorithm B , and then we bound the advantage of the adversary. The arbiter algorithm B operates as follows:

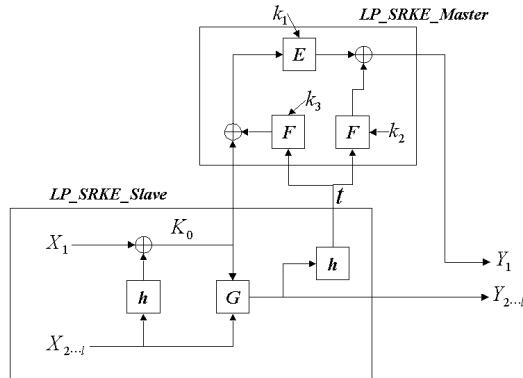


Fig. 5. LP_SRKE encryption protocol

The arbiter algorithm B

For $i \in \{1, \dots, q_h\}$, B compiles a list L_1 of all the pairs (K_0^i, t^i) and another list L_2 of all (Y_1^i, t^i) . These can be deduced from the transcript. If an attacker A asks for the encryption of a plaintext X^j with $j \in \{q_h + 1, \dots, q\}$ in the distinguishing phase, the first challenge value for the card is the pair (K_0^j, t^j) where $K_0^j = X_1^j \oplus h(X_2^i, \dots, X_l^i)$. If the pair (K_0^j, t^j) is contained in the list L_1 , B acts on that query, and the answer is generated according to the encryption protocol. Similarly, if A asks for the decryption of a ciphertext Y^j in the distinguishing phase, and if the corresponding challenge value (Y_1^j, t^j) is contained in L_2 , B acts and the answer is generated according to the decryption protocol. B does not act on more than q_h queries. Due to the collision resistance of h , A does not know more than one value (X_2^i, \dots, X_l^i) with $K_0^i = X_1^i \oplus h(X_2^i, \dots, X_l^i)$. For the same reason, A knows only one value (Y_2^i, \dots, Y_l^i) with $t^i = h(Y_2^i, \dots, Y_l^i)$. Hence, for every $i \in \{1, \dots, q_h\}$, A can ask no more than one plaintext X^j to be encrypted during the distinguishing phase, where the corresponding pair (K_0^j, t^j) would be found in the list L_1 . Similarly, we argue for decryption queries. Asking for the encryption of $T = (X_1^j, X_2^j, \dots, X_l^j)$ is equivalent to asking for the decryption of $T' = (Y_1^j, Y_2^j, \dots, Y_l^j)$, and we only regard non-equivalent queries. ♣

Let $k > q_h$ and A be asking for the encryption of a plaintext $(X_1^k, X_2^k, \dots, X_l^k)$. We assume $(K_0^k, t^k) \notin \{(K_0^1, t^1), \dots, (K_0^k - 1, t^k - 1)\}$. If $j \in \{1, \dots, q_h\}$ and $(K_0^j, t^j) = (K_0^k, t^k)$, then B acts and the answer to this query does not depend on the switch S . If $j \in \{q_h + 1, \dots, k - 1\}$, then $(X_1^j, X_2^j, \dots, X_l^j) \neq (X_1^k, X_2^k, \dots, X_l^k)$, because the j -th and the k -th query are not equivalent. If $(X_1^j, X_2^j, \dots, X_l^j) \neq (X_1^k, X_2^k, \dots, X_l^k)$, then $(K_0^j, t^j) = (K_0^k, t^k)$ implies a collision $(X_2^j, \dots, X_l^j) \neq (X_2^k, \dots, X_l^k)$ for h .

Depending on previous protocol runs and responses, we define the set $U_k \subseteq \{0, 1\}^b \times \{0, 1\}^{\beta-b}$ of ciphertexts (β is a length of a plaintext):

$$U_k := (\{0, 1\}^b - \{Y_1^1, \dots, Y_l^{k-1}\}) \times (\{0, 1\}^{\beta-b})$$

For $S = 1$, the ciphertext $(Y_1^j, Y_2^j, \dots, Y_l^j)$ is a uniformly distributed random value in $\{0, 1\}^b \times \{0, 1\}^{\beta-b}$. We define what it means for the k -th query to be Σ_k , $\Sigma_k \Leftrightarrow Y_1^k \in \{Y_1^1, \dots, Y_l^{k-1}\}$. Obviously, if not Σ_k , then $(Y_1^k, Y_2^k, \dots, Y_l^k)$ is a uniformly distributed random value in U_k . Further:

$$P[\Sigma_k | S = 1] \leq \frac{k-1}{2^b}$$

Consider $S = 0$. Obviously, if not Σ_k , then $(Y_1^k, Y_2^k, \dots, Y_l^k) \in U_k$. Also, if not Σ_k , then K_0^k is not in $\{K_0^1, \dots, K_0^{k-1}\}$, and then $G_{K_0^k}(Y_2^k, \dots, Y_l^k)$ is a uniformly distributed random value in $\{0, 1\}^{\beta-b}$. Furthermore, if $t^j = t^k$, $Y_1^j \neq Y_1^k$ due to $K_0^j \neq K_0^k$.

If $K_0^j \neq K_0^k$, then $E_{k_1}(K_0^j)$ and $E_{k_1}(K_0^k)$ are two independent random values in $\{0, 1\}^b$. If $(K_0^k, t^k) \notin \{(K_0^1, t^1), \dots, (K_0^{k-1}, t^{k-1})\}$, for every $j \in \{1, \dots, k-1\}$, $K_0^j \neq K_0^k$ if $t^j = t^k$. Hence, $P[Y_1^j = Y_1^k | K_0^j \neq K_0^k] \leq 2^{-b}$.

$$P[\Sigma_k | S = 0] \leq \frac{k-1}{2^b}$$

We write Σ^* if any query in k in the distinguishing phase is Σ_k .

$$\begin{aligned} P[\Sigma^* | S = 1] &\leq \sum_{k=q_h+1}^q P[\Sigma_k | S = 1] \leq \frac{1}{2} \frac{q^2}{2^b} \\ P[\Sigma^* | S = 0] &\leq \sum_{k=q_h+1}^q P[\Sigma_k | S = 0] \leq \frac{q^2}{2^b} \end{aligned}$$

If A asks for the decryption of the ciphertext $(Y_1^k, Y_2^k, \dots, Y_l^k)$ as the k -th query instead of asking for an encryption, the same argument applies. Therefore, the advantage of A is

$$\text{Adv}_A \leq \frac{q^2}{2^b}$$



We compare the proposed LP-SRKE protocol with BFN's length-preserving RKE and Lucks's ARKE. BFN's scheme and ARKE operate as Fig.6 and Fig.7, respectively. On the part of the host, the cryptographic calculations are exactly the same for three protocol. While BFN's scheme and ARKE require two interactions between the host and the card, LP-SRKE requires only one interaction. Inside the card, BFN's scheme requires six evaluations of cryptographic functions and ARKE requires four evaluations. However, LP-SRKE requires three evaluations of cryptographic functions. Therefore, the proposed LP-SRKE has smaller computations and requires only one interaction between the host and the card. This scheme can be a solution for the open problem suggested by BFN[4][5]:

Is there a secure, length-preserving RKE that requires only one round of interaction?

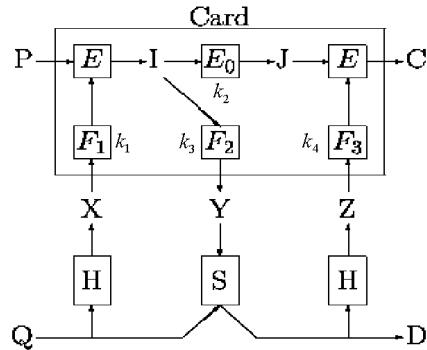


Fig. 6. BFN's length-preserving RKE protocol

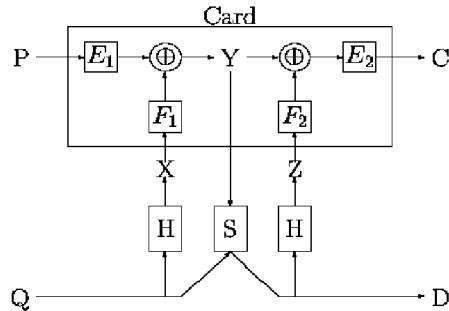


Fig. 7. ARKE protocol

5 Conclusions

In this paper, we proposed new secure remotely keyed encryption protocols; a secure, length-increasing RKE and a secure, length-preserving RKE. The proposed protocols have smaller computations than the existing RKE protocols and provide sufficient security. Especially, because the proposed length-preserving RKE requires only one interaction between the host and the card, we suggested the proposed length-preserving scheme as the solution for the open problem (*Is there*

a secure, length-preserving RKE that requires only one round of interaction?) suggested by BFN. Also, as a cryptographic function G in the host of the proposed protocols, we used the IAON_G with all-or-nothing properties proposed by Rivest. The IAON_G is an improved version of AON hashing-3 proposed by Shin, et al[11], and provides better security without additional computation overhead.

References

1. M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication," in Advances in Cryptology - Crypto'96, LNCS, vol.1109, pp.1-15, 1996 [182](#)
2. M. Bellare, J. Kilian, R. Rogaway, "The Security of Cipher Block Chaining," in Advanced in Cryptography - Crypto'94, LNCS, vol.839, Springer, pp.341-358, 1994 [187](#)
3. M. Blaze, "High-Bandwidth Encryption with Low-Bandwidth Smartcards," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1039, Springer, pp.33-40, 1996 [183, 184](#)
4. M. Blaze, J. Feigenbaum, M. Naor, "A Formal Treatment of Remotely Keyed Encryption," in Advanced in Cryptography - Eurocrypt'98, LNCS, vol.1403, pp.33-40, 1998 [179, 184, 185, 191, 193](#)
5. M. Blaze, J. Feigenbaum, M. Naor, "A Formal Treatment of Remotely Keyed Encryption," Full version of Eurocrypt'98, 1999 [179, 184, 185, 186, 188, 190, 191, 193](#)
6. M. Jakobsson, J.P. Stern, M. Yung, "Scramble All, Encrypt Samll," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1636, 1999 [183, 184](#)
7. S. Lucks, "On the security of Remotely Keyed Encryption," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1267, Springer, pp.219-229, 1997 [184](#)
8. S. Lucks, Accelerated Remotely Keyed Encryption," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1636, 1999 [184, 185, 191](#)
9. B. Preneel, P. van Oorschot, "MDx-MAC and Building Fast MACs from Hash Functions," in Advances in Cryptology - Crypto'95, LNCS, vol.963, pp.1-14, 1995 [182](#)
10. R. L. Rivest, "All-Or-Nothing Encryption and The Package Transform," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1267, pp.210-218, 1997 [179](#)
11. Sang Uk Shin, Kyung Hyune Rhee, Jae Woo Yoon, "Hash Functions and the MAC Using All-Or-Nothing Property," in Proceedings of PKC'99(International Workshop on Practice and Theory in Public Key Cryptography), LNCS, vol.1560, pp.263-275, 1999 [179, 180, 195](#)

Security of Public Key Certificate Based Authentication Protocols

Wu Wen, Takamichi Saito, and Fumio Mizoguchi

Science University of Tokyo, 2641 Yamazaki, Noda, Chiba, Japan 278-8510
{wen,mizo}@imc.sut.ac.jp, saito@yu.is.noda.sut.ac.jp

Abstract. The security of authentication protocols based on public key cryptography depends on the validity of the certificate. It is usually assumed that a well deployed PKI can guarantee the validity of certificates through mechanisms such as CRL or OCSP. In reality, such guarantee is not always assured. This paper describes an attack that exploits this certificate validity weakness and breaks some well-known certificate-based authentication protocols, namely the SSL and the TLS protocol. This attack affects the “named-server” version of both protocols, but is ineffective for the “named-server, named-client” version of both protocols. Along with the attack, we also describe how it was discovered as a result of our ongoing research on analysis of authentication protocols using both logic based and model checking based methods.

1 Introduction

Soon after the public-key cryptography[3] was invented a rather simplistic approach towards how it can be used to provide integrity and security for digital messages was proposed: a yellow page with name and public key pairs is to be distributed just like a telephone book. To make sure names and public keys are securely binded, certificates[7] were introduced. Later on, the concept of a Public Key Infrastructure(PKI) is envisioned so that the validity of all certificates can be verified by the possession of the root verification key. To solve the problem that valid certificates can become invalid in time due to various reasons, a Certificate Revocation List(CRL) is also provided in PKI.

In theory, any principal who believes that his certificate is compromised can revoke that certificate by putting it on the CRL. Principals intending to use certain certificate can check its validity by searching the CRL. More recently, to lessen the problem of having the client to comb through all the CRLs before knowing if a certificate is invalid, Online Certificate Status Protocol(OCSP) is proposed. These measures, however, can not provide absolute guarantee for certificate validity at all times. On the other hand, commercial products such as the popular Netscape and the Explorer browser have incorporated certificate based authentication protocols such as the Secure Socket Layer(SSL) and Transport Layer Security(TLS). The security of public key certificate based authentication protocols in less ideal situations where a certificate may have been compromised needs to be further analyzed.

Authentication is usually conducted between two parties A and B . When public key cryptography is used, public key certificate plays a critical role in authentication protocols thus constructed. A certificate $\{A, K_A^+\}_{K_{CA}^-}$ binds a name A of a principal with its public key K_A^+ through signing using the signature key K_{CA}^- of a trusted third party CA , also known as the Certificate Authority. The secrecy of a message M encrypted by this public key, $\{M\}_{K_A^+}$ is guaranteed by the fact that only the principal with the corresponding private key K_A^- can decrypt this message. In addition, the integrity of a message M signed with the signature key K_A^- , $\{M\}_{K_A^-}$, can be verified with the corresponding verification key K_A^+ .

Let us first compare the consequences as a result of key compromises in symmetric shared secret based and public key based systems. In a symmetric shared secret based authentication system, such as a password based authentication system, if the password is compromised, and the fact that it is compromised is not known to both parties of the authentication run, the system is considered broken. Similarly, if the private key corresponding to a certain certificate is compromised, and the fact that it is compromised is not known to both parties, the authentication protocol based on the validity of the certificate is considered broken. This paper does not concern such trivial situations.

In this paper, we are concerned with situations in which a certificate of certain principal is *known* to have been compromised, and is revoked immediately. In the password based authentication example, if the password compromise is known, the promised password will be revoked and a new password will be used. The security of the system is then considered restored. Could this be said of the public key certificate based authentication protocols? This paper describes an attack that exploit the “known” compromised certificate situation and render some version of the TLS and SSL insecure. Furthermore, the ultimate aim of many authentication protocols is to establish a master session key between the authenticated parties to conduct further secure communication. For example, the Netscape browser allows its users to authenticate a web server using SSL before the user gives out his credit card number. The master session key thus generated is then used to encrypt sensitive communication such as the credit card number. In this paper we are concerned with the possibility of leaking of this master session key without the private key of the server certificate currently under use.

2 The Attack on TLS Protocol

In this section, we first give a concrete description of the “named-server, anonymous-client” version of the the TLS protocol. This is followed by a problem statement giving the assumptions made about the environment and the problem we are addressing. Finally, detailed attack trace is presented with discussion of its implications.

2.1 The TLS Protocol Description

In the following, we give a brief description of the TLS protocol by listing the messages communicated between the client and the server. The basic mode of the TLS protocol is the “named-server, anonymous-client” version of the protocol, in which only the server, such as an internet shopping mall, is required to have a certificate. In addition to authenticate the server, this protocol also generates a master session key between an anonymous client and a named server. We use the following notations:

C	Client
S	Server
CA	Certificate Authority
T_i	Timestamp generated by principal i
N_i	Random nonce generated by principal i
K_i^+	Public encryption key for principal i
K_i^-	Secret key of principal i
$\{i, K_i^+\}_{K_{CA}^-}$	i 's Public key certificate
$\{\dots\}_{K_i^-}$	Signed by principal i with key K_i^-
$\{\dots\}_{K_i^+}$	Encrypted with principal i 's public key K_i^+

According to [4,2], the six messages of the protocol are as follows:

$$\begin{aligned} C \rightarrow S : (N_C, T_C) & \quad (M_1) \\ S \rightarrow C : (N_S, T_S) & \quad (M_2) \\ S \rightarrow C : \{S, K_S^+\}_{K_{CA}^-} & \quad (M_3) \\ C \rightarrow S : \{N'_C\}_{K_S^+} & \quad (M_4) \\ S \rightarrow C : \{H(K_{AB}, AB_5, (M_1, M_2, M_3, M_4))\}_{K_{AB}} & \quad (M_5) \\ C \rightarrow S : \{H(K_{AB}, AB_6, (M_1, M_2, M_3, M_4))\}_{K_{AB}} & \quad (M_6) \end{aligned}$$

where

$$K_{AB} = F((N_C, T_C, N_S, T_S), N'_C)$$

and

$$AB_5 = \text{"server finished"}, AB_6 = \text{"client finished"}$$

The messages can be summarized as follows:

- M_1 : C sends a timestamp and a nonce to S ;
- M_2 : S sends a different timestamp and nonce to C ;
- M_3 : S sends its certificate to C ;
- M_4 : C returns the “pre-master secret” N'_C encrypted under K_S^+ ;
- M_5 : S sends a hash of the session key, a tag AB_5 indicating the protocol stage, and all preceding messages sent by S to C , encrypted with session key K_{AB} ;
- M_6 : C sends a hash of the session key, a tag AB_6 indicating the protocol stage, and all preceding messages sent by C to S , encrypted with session key K_{AB} .

The master session key K_{AB} is also used by the record layer to encrypt all communications from this point on.

2.2 Problem Statement

In the following, we define the problem we are addressing by stating the following assumptions:

Certificate validity assumption:

- A certificate is always verified using the verification key of the CA. It is impossible for a third party to fake such certificate;
- Certificates have unique names. It is impossible for a client to mistake a third party's certificate for that of the server;
- The signature key of the CA is secure;
- CRLs are only published periodically. It is reasonable to assume that a revoked certificate may still be used before the user finds out from either the CRL, or the owner of the certificate that it is revoked;
- OCSP is not integrated with TLS, the validity of the certificate is not always checked at the start of the TLS session.

The last two assumptions appear to be artificial. However, CRL's inadequacies have been pointed out by various researchers[5,6]. We would further argue that

- in the CRL scheme, although the integrity of the CRL is assured by CA's verification key, it is not scalable if the client must keep track of all the related CRLs;
- in the OCSP scheme, a secure channel must be established between the enquiring client and the OCSP server to assure its integrity. However, to establish such a secure channel, protocols like TLS are needed. This leads to a chicken-and-egg scenario;
- if TLS includes certificate status information in message M_3 , before a secure connection is established between the server and the client, such status information may not be trusted.

It is apparent from the above arguments that there is no easy way for a client to have correct and timely certificate status information at all times. We further assume the circumstances of a compromised certificate as follows:

Compromised certificate assumption:

- a third party has obtained a copy of the private key that corresponds to the server's certificate;
- the owner of the certificate is aware of the compromise;
- the owner revokes the certificate and lists it on the CRL,
- and the owner obtains a new certificate from the CA.

The above assumptions describe a plausible scenario in today's internet-based use of certificates such as those involving SSL with Netscape and Explorer

browsers. For example, if a webmaster for an on-line banking site is fired, the bank will assume that the corresponding certificate is compromised. If the bank is sensible, it is likely to revoke the certificate and obtain a new certificate for future use.

If we ignore the trivial case where the ex-webmaster simply sets up a fake website to impersonate the on-line banking site, we may intuitively argue that the secrecy requirement of TLS seems to be satisfied. Let us assume that the intruder I replaces message M_3 in the TLS protocol, described in section 2.1, with the compromised certificate. Upon receiving the compromised certificate, C either find out from CRL that it is compromised, or send M_3 using the public key contained in the compromised certificate. On receiving M_4 , S will find out that it is unable to decrypt it since he now has a new secret key corresponding to the new certificate. The protocol run will be aborted and C will then obtain the new certificate. It may be argued that since N'_C is just a random number, S will simply decrypt M_4 without noticing it was encrypted with a different key. In that case, the calculated master secret K_{AB} will be different for C and S and the protocol will be aborted after message M_5 . In this case, the TLS server is in fact also acting as an OCSP server in showing that the certificate used by the client is invalid. In the following we show such arguments do not stand.

2.3 How to Stealing the Master Secret

In the following, we describe the steps showing how an intruder with a compromised certificate can learn the pre-master secret, and therefore the master session key, even if the server has revoked and updated its certificate. Message in the form of “ $\dots \rightarrow (X)I$ ” indicates that a message intended for X is intercepted by I . Message in the form of “ $(X)I \rightarrow \dots$ ” indicates a message faked by I as from X . Note that $\{S, K_S'^+\}_{K_{CA}^-}$ is the compromised certificate of S while $\{S, K_S^+\}_{K_{CA}^-}$ is the fresh and valid certificate of S . M'_i and M''_i are messages that are intercepted and faked by the intruder respectively. They are identical in format with M_i otherwise.

$$\begin{aligned}
C \rightarrow S : & (N_C, T_C) & (M_1) \\
S \rightarrow C : & (N_S, T_S) & (M_2) \\
S \rightarrow (C)I : & \{S, K_S^+\}_{K_{CA}^-} & (M'_3) \\
I(S) \rightarrow C : & \{S, K_S'^+\}_{K_{CA}^-} & (M''_3) \\
C \rightarrow (S)I : & \{N'_C\}_{K_S'^+} & (M'_4) \\
(C)I \rightarrow S : & \{N'_C\}_{K_S^+} & (M''_4) \\
S \rightarrow (C)I : & \{H(K_{AB}, AB_5, (M_1, M_2, M'_3, M''_4))\}_{K_{AB}} & (M'_5) \\
(S)I \rightarrow C : & \{H(K_{AB}, AB_5, (M_1, M_2, M''_3, M'_4))\}_{K_{AB}} & (M''_5) \\
C \rightarrow (S)I : & \{H(K_{AB}, AB_6, (M_1, M_2, M''_3, M'_4))\}_{K_{AB}} & (M'_6) \\
(C)I \rightarrow S : & \{H(K_{AB}, AB_6, (M_1, M_2, M'_3, M''_4))\}_{K_{AB}} & (M''_6)
\end{aligned}$$

In the following we briefly explain the meaning of some of the above messages.

- M_1 and M_2 are sent in plain text;

- In M'_3 the valid certificate for S is intercepted;
- The intruder substitutes it with the compromised certificate;
- In M'_4 , the intruder intercept the message and learns N'_C , the pre-master secret;
- The intruder then fake M'_4 using the public key contained in the valid certificate;
- Since the intruder knows N_C, T_C, N_S, T_S and N'_C , it is able to calculate the master session key K_{AB} ;
- The interception and faking of the last four messages are now possible and required since both C and S must maintain a consistent record of the past messages despite the different versions of M_3 and M_4 kept by C and S respectively.

Note that messages M_5 and M_6 were designed to prevent attacks that are based on interception and faking of messages M_1, \dots, M_4 . For example, if M'_5 is allowed to reach C , C will find out that the hash of all messages doesn't match because C expects (M_1, M_2, M''_3, M'_4) but instead receiving (M_1, M_2, M'_3, M''_4) . We can see from above that it is ineffective in preventing the interception and faking described above because the master session key K_{AB} is known to the intruder. After the verification steps M_5, M_6 , a TLS session is regarded securely established between C and S . Unfortunately the master secret is now known by a third party. Credit card number and pin numbers that are sent over this TLS session is now known to the third party, an ex-webmaster for example.

2.4 Discussion

TLS is derived from SSL. In the “named-server” version of the SSL protocol, identical attack as describe above also leads to the leaking of the the master session key K_{AB} . However, in the “named-sever, named client” version of both SSL and TLS protocol, the above attack doesn't succeed. This is explained next.

In the “named-sever, named-client” version of the TLS protocol, both clients and server are authenticated. In message M_4 of this version of the protocol, C 's certificate and a signed hash of the list of previous messages so far transmitted, in addition to the pre-master secret encrypted with S 's public key, are sent by C to S . This is shown as follows:

$$C \rightarrow S : \{C, K_C^+\}_{K_{CA}^-}, \{N'_C\}_{K_S^{'+}}, \{\dots, \text{Hash}(M_1, M_2, M''_3), \dots\}_{K_C^-}(M_4)$$

The parts shown as “...” are tags and other parameters such as the master secret computed by C at this time and are not important for the following discussion. Again, the intruder I can learn the pre-master secret N'_C using $K_S^{'+}$, and compute the master secret K_{AB} . It is however, unable to fake the signed portion of M_4 because it does not have the signature key K_C^- of the client C . In order to maintain the consistency of the list of transmitted messages at both C and S side, the signed part, as well as the encrypted pre-master secret in M_4 , must also be altered. Due to the inability to alter the signed portion of M_4 by the intruder, it is possible now to detect the interception and faking of M_3 and M_4 in step M_5 or M_6 . The protocol run will be aborted and the attack will fail.

3 How Were the Above Attacks Discovered?

We have been interested in using both model checking method[14] and logic checking method[13] for analyzing authentication protocols. First we extended the BAN logic[1] to allow it to deal with the analysis of secrecy theorems and succeeded in discovering a weakness in the fix[8] proposed by Gavin Lowe for the Needham-Schroeder public key authentication protocol. Model checking is then used to discover the exact attack for the described weakness. It turned out that the weakness discovered in the Needham-Schroeder protocol affects the “named-server” versions of both SSL and TLS.

The next two sections describe in summary the results of our analysis of the Needham-Schroeder authentication protocol using both methods. Readers interested in the details should consult [13] and [15].

4 Extending BAN Logic for Secrecy Analysis

BAN logic was proposed to reason and analyze properties of authentication protocols such as Kerberos, Needham-Schroeder, and SSL etc. A number of weaknesses in various protocols were discovered using BAN logic. However, due to its modeling of the principals as benign agents, as pointed out by Dan Nessett in [10], it is considered ineffective in analysis of secrecy related properties of authentication protocols. As described in [10], the environment modeled in BAN does not include principals that are not supposed to gain access to certain secrets. Since such principals are not modeled, it is therefore impossible to analyze property related to secrecy. Nessett further illustrated this weakness by describing an obviously flawed protocol and proving proper authentication using the BAN logic.

The criticism is somewhat unfair in that, the designer of BAN logic has never claimed it can be used to deal with secrecy. Only proper authentication, as defined by a set of beliefs, were defined and used in the analysis.

Our work is partially motivated by this criticism, as well as other recent research in dealing with the analysis of secrecy properties of authentication protocols. For example, Paulson has incorporated secrecy theorems into his logic for security protocols in [11]. Roscoe described a model for a “spy” in [12]. Lowe has successfully discovered an attack[8] on Needham-Schroeder public key authentication protocol using a model of the protocol including an intruder. Other results on verification of SSL[9] and TLS[4,11] also encouraged our investigation.

4.1 Parameterization of BAN Logic

In the following, the Needham-Schroeder public key authentication protocol is described briefly before the parameterization is explained using this protocol as an example. We will refer to this protocol as the original protocol from now on. In the simplified version of this protocol, only two principals, an initiator A and a responder B are considered. Three messages are exchanged:

$$\begin{aligned} A \rightarrow B : \{N_A, A\}_{K_B^+} & \quad (M_1) \\ B \rightarrow A : \{N_A, N_B\}_{K_A^+} & \quad (M_2) \\ A \rightarrow B : \{N_B\}_{K_B^+} & \quad (M_3) \end{aligned}$$

We introduce a third principal, an intruder I and model the above protocol using parameterization. First we fix the initiator A , then we parameterize the responder by a variable X_B , where $X_B \in \{B, I\}$. This allow us to model the protocol run that includes possible interception and faking of messages by the intruder. For example, the following message:

$$A \rightarrow X_B : \{N_A, A\}_{K_{Y_B}^+}, \text{ where } Y_B \in \{B, I\},$$

can be interpreted as

- $X_B = Y_B = B$: a message sent from A to B encrypted with B 's public key, which corresponds to the normal message described in the original protocol definition;
- $X_B = Y_B = I$: a message sent from A to I using I 's public key;
- $X_B = B, Y_B = I$: a message sent from A , intended for B , using I 's public key.

Note that we use Y_B and X_B to indicate that the intended receiver and its public key bearer may be different.

4.2 Secrecy Property in Parameterized BAN

With the introduction of an intruder I , we are in the position to define secrecy property for the above protocol.

Definition 1. *If both shared secret N_A and N_B , intended for an authentication session between A and B , can be obtained by intruder I , the secrecy property of the protocol is considered violated. In terms of the parameterized BAN logic, if the formulas corresponding to “ I saw N_A ” and “ I saw N_B ” can be derived from the protocol, we regard the secrecy of the protocol violated.*

For the original Needham-Schroeder protocol described above, both formula corresponding to “ I saw N_A ” and “ I saw N_B ” can be derived[13]. This indicates that the secrecy property for the protocol is violated. Unfortunately, the analysis does not directly point to an attack. In [8], Lowe discovered the following attack using model checking method. We will refer to this attack as the “impersonation attack”.

$$\begin{aligned} \alpha : A \rightarrow I & \quad \{N_A, A\}_{K_I^+} \\ \beta : (A)I \rightarrow B \{N_A, A\}_{K_B^+} & \\ \beta : B \rightarrow (A)I \{N_A, N_B\}_{K_A^+} & \\ \alpha : I \rightarrow A & \quad \{N_A, N_B\}_{K_A^+} \\ \alpha : A \rightarrow I & \quad \{N_B\}_{K_I^+} \\ \beta : (A)I \rightarrow B \{N_B\}_{K_B^+} & \end{aligned}$$

The above trace describes two interleaving runs α and β . Run α is between A and I , and run β is between $(A)I$ and B . $(A)I$ indicates intruder I impersonates A . Lowe proposed a fix in [8] to deny this attack. In the fixed protocol, message 2 is

$$B \rightarrow A : \{N_A, N_B, B\}_{K_A^+}$$

We will refer to this addition as “Lowe’s fix”. This prevents the “impersonation attack” since message 2 can not be simply copied from run β to run α . A will expect I rather than B as its corresponded. This fix, however, is shown ineffective to the “ex-webmaster attack” described in the section 5.4.

4.3 Security of the Fixed Needham-Schroeder Protocol

We idealize message 2: $\{N_A, N_B, B\}_{K_A^+}$ in the fixed Needham-Schroeder protocol by adding the substitution term: $X_B = B$ to our model of the Needham-Schroeder protocol using parameterized BAN logic. In this case, the inference results does not contain the formula “ I see N_B ”. Formulas corresponding to proper authentication described in the original BAN logic can also be obtained.

This result appears to indicate that the fixed Needham-Schroeder is secure. However, on closer examination, the secrecy property is held if the assumption $X_B = B$ is guaranteed. The security of the fixed protocol is therefore dependent on this assumption. This turned out to be the weakness of the fixed Needham-Schroeder protocol.

Quick discussion: Analysis based on logic, such as the BAN logic, does not directly point to possible attacks. Rather, it highlights the weakness in the protocol by pointing to certain weak assumptions that may not stand in reality. Further analysis of such assumptions may lead to the discovery of actual attack. Our study also follows this pattern. In the next section, we show how an attack on this fixed protocol can be discovered.

5 Discover Concrete Attacks Using Model Checking

Unlike logic based methods such as the BAN logic, in which messages are abstracted and idealized to logic formulas representing certain beliefs held by each principals, model checking method requires a concrete state based model to be constructed. Usually, each principal is represented by a process. States of the process such as “running” or “committed” are reached depending on the messages that are sent and received. If a specification described as an assertion or a temporal logic formula is shown to be untrue with respect to the model, the model checker also provides a trace of the actual attack.

In the following, we describe the state based model of the protocol that includes an initiator, a responder and an intruder. The secrecy requirements used to discover the “ex-webmaster attack” is also described.

5.1 State-based Model of Needham-Schroeder Protocol

The state-based model consists of three concurrent processes representing the initiator, the responder and the intruder. The initiator and the responder send and receive messages strictly according to the protocol specification. The intruder, however, are allowed the following behavior:

- overhear and store a copy of the message;
- steal/intercept a message from its intended receiver;
- decrypt message that are encrypted with its public key;
- replay any stored message at any time;
- make up new messages using learned secret such as stolen nonces.

The above assumptions are based on those describe in [8] and are justified in the current Internet environment assuming that cryptography used by the protocol is strong enough. An illustration of the model is shown in Figure 1.

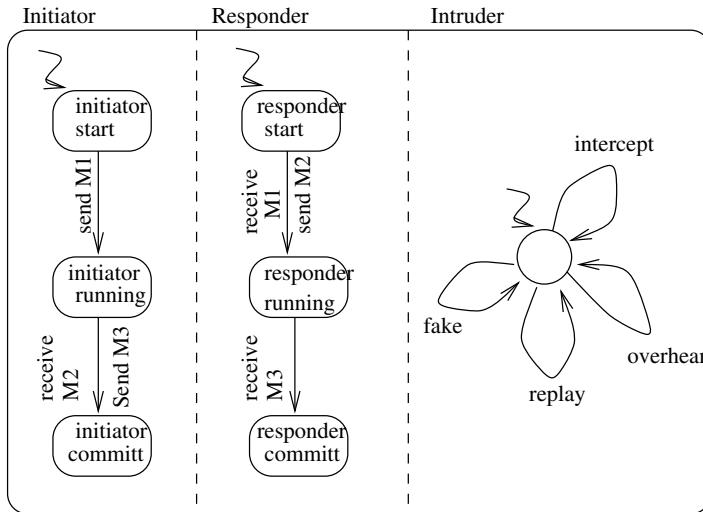


Fig. 1. The state-based model for Needham-Schroeder Protocol

Based on the results describe in section 4.3, it is clear that the security of Lowe's fix depends on whether we can guarantee $X_B = B$ when receiving message 2. This is assumed in Lowe's analysis. To further explore the consequences of possible certificate compromises we deliberately weaken this assumption and allow the possibility that A may not have the means to positively identify X_B as B . In other words, it is possible for A to send a message intended for B using public key that may or may not be that of B 's. This is to say that message of the following format

$$A \rightarrow B : \{N_A, A\}_{K_I^+}$$

is not excluded from our model.

This is the only difference between our protocol model and that of Lowe's. We will give an justification for weakening this assumption in section 5.4.

5.2 Nonce Secrecy Requirements

In [8], the requirements for proper authentication are described as safety properties:

- If the responder is committed to the initiator, the initiator must be running with the responder;
- If the initiator is committed to the responder, the responder must be running with the initiator.

It is quite clear that such requirements are concerned with the possibility of the intruder impersonating either the initiator or the responder.

Our results of analysis given in logic checking phase, however, points to the necessity of protecting the secrecy of the nonces. Consequently, our requirements are given as a safety property of the secrecy of the nonces N_A and N_B .

- Intruder I must not be able to learn both N_A and N_B in protocols runs between A and B .

5.3 Results of Model Checking

Two versions of the Needham-Schroeder protocol were model checked using the secrecy requirement described in the previous subsection: the original Needham-Schroeder and the version with Lowe's fix.

Both the “impersonation attack” and the “ex-webmaster attack” are found in the results of model checking the original version of the protocol. Only the “ex-webmaster attack” was discovered in the version with Lowe's fix. The traces of the attacks are shown in Figure 2 and Figure 3 respectively. Note that the “impersonation attack” is discovered using the secrecy requirement rather than the proper authentication requirement used by Lowe. In fact, Lowe's attack can also be described as the result of I learning N_A, N_B , which is supposed to be a shared secret between A and B .

The model checking result for the version with Lowe's fix only produced the attack corresponding to the “ex-webmaster attack”. In other words, Lowe's fix is effective towards the “impersonation attack” but not effective towards the “ex-webmaster attack”. The author can verify this by modifying message 3 in Figure 3 and see that it has no effects in preventing the “ex-webmaster attack”.

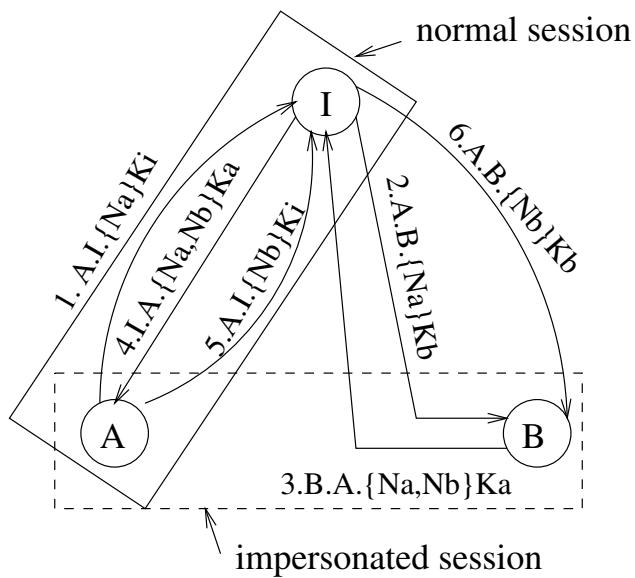


Fig. 2. The “impersonation attack” due to Lowe

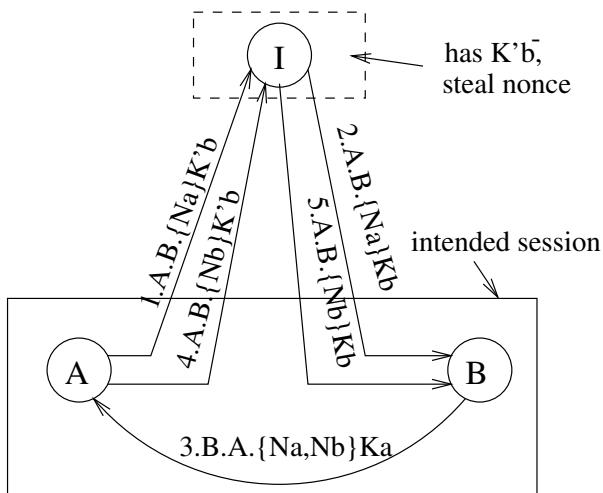


Fig. 3. The “ex-webmaster attack” due to Wen-Saito

5.4 Interpretation of the “Ex-Webmaster Attack”

On first glance, it might be difficult to imagine that A might, by its own will, send an encrypted message addressed to B , using the public key that belongs to the intruder. Closer analysis shows that such situation is quite possible. As described in section 2.2 as the attack on the “named-server” version of both SSL and TLS protocol, the client C sends the pre-master secret N'_C using the compromised public key of the server S . We can easily suggest that the compromised public key of the server, is in fact the public key of the intruder described in here.

5.5 Another Fix to Needham-Schroeder

We propose a fix that can eliminate both “impersonation attack” and the “ex-webmaster attack” on the public key version of the Needham-Schroeder authentication protocol. Instead of adding only B to message 2 in the original protocol, we propose to add K_B^+ . The fixed protocol is as follows:

$$\begin{aligned} A \rightarrow B : \{N_A, A\}_{K_B^+} & \quad (M_1) \\ B \rightarrow A : \{N_A, N_B, K_B^+\}_{K_A^+} & \quad (M_2) \\ A \rightarrow B : \{N_B\}_{K_B^+} & \quad (M_3) \end{aligned}$$

Since the intruder I is unable to decrypt message 2 and replace K_B^+ with $K_B'^+$, A will be able to detect the inconsistency. Even if it chooses not to check for inconsistency of K_B^+ and $K_B'^+$, message 3 will be encrypted with K_B^+ , the new and valid public key, and the intruder will be unable to learn nonce N_B from message M_3 .

6 Discussion

When analyzing certificate based protocols, it is normally assumed that the validity of the certificates is guaranteed by the deployed PKI. In reality, methods for guaranteeing such validity, such as CRL and OCSP, are either ineffective, or difficult to be incorporated into the various authentication protocols used currently. The case of total loss of security as the result of a compromised certificate, where the owner is not aware, does not seem to have a solution. If, however, one of the principals engaged in the authentication run has a compromised certificate, knows about it, revokes it and replaces it with a new certificate, it appears that the protocol should be safe since the owner will detect such inconsistency and stop the run. The results described in this paper shows that an attack taking advantage of the “known” invalid certificate exists for the “named-server” version of the most popular authentication protocols used today.

The ineffectiveness of the attack towards the “named-server, named-client” version of both SSL and TLS suggest that, signing the hash of all previous messages by the client has its merit.

Unfortunately, the fix we proposed for the Needham-Schroeder public key authentication protocol is unsuitable for fixing SSL and TLS. Since the client

certificate is not used in the “named-sever” only version of SSL and TLS, it is not possible to securely pass the new and valid certificate in the current protocol run. If client C ’s certificate is used, as in the “named-server, named-client” version of the protocol, signing the hash of all previous messages by the client should also prevent this attack.

References

1. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report 39, DEC Systems Research Center, February 1989. [202](#)
2. T. Dierks and C. Allen. The tls protocol: Version 1.0. Technical Report drat-ietf-tls-rptocoll-05.txt.Z, IETF task force, May 1998. [198](#)
3. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976. [196](#)
4. Walter D. Eaves. Transport level security: a proof using the gny logic. Technical report, Brunel University, UK, February 1989. [198](#), [202](#)
5. C. Ellison. Can we eliminate crl? Technical report, extended abstract, November 1997. [199](#)
6. C. Ellison and et al. Spki certificate theory, internet draft. Technical report, IETF SPKI Working Group, November 1997. [199](#)
7. L. Kohnfelder. Towards a practical public-key cryptosystem. Technical report, MIT, 1978. [196](#)
8. G. Lowe. Breaking and fixing the needham-schroeder public key protocol using csp and fdr. In *TACAS96*, 1996. [202](#), [203](#), [204](#), [205](#), [206](#)
9. J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murphi. In *IEEE Symposium on security and privacy*, pages 141–151, 1997. [202](#)
10. D. Nessett. A critique of the burrows, abadi and needham logic. *ACM Operating Systems Review*, 24(2):35–38, April 1990. [202](#)
11. L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998. [202](#)
12. A. W. Roscoe. The perfect ‘spy’ for model-checking crypto protocols. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, Piscataway, NJ, September 1997. [202](#)
13. T. Saito, W. Wen, and F. Mizoguchi. Analysis of authentication protocol by parameterized ban logic. Technical report, ISEC, July 1999. [202](#), [203](#)
14. W. Wen and F. Mizoguchi. Model checking security protocols. In *Symposium on Cryptography and Information Security*, pages 647–652, Kobe, Japan, January 28–29, 1999. [202](#)
15. W. Wen, T. Saito, and F. Mizoguchi. New results of model checking needham-schroeder protocol. in proceedings of 1999 Computer Security Symposium, pages 69–74, Kanazawa, Japan, October 21–22, 1999. [202](#)

Efficient Implementation of Schoof's Algorithm in Case of Characteristic 2

Tetsuya Izu, Jun Kogure, and Kazuhiro Yokoyama

FUJITSU LABORATORIES LTD., 4-1-1 Kamikodanaka

Nakahara-ku Kawasaki 211-8588, Japan

{izu, kogure}@flab.fujitsu.co.jp, yokoyama@sec.flab.fujitsu.co.jp

Abstract. In order to choose a secure elliptic curve for Elliptic Curve Cryptosystems, it is necessary to count the order of a randomly selected elliptic curve. Schoof's algorithm and its variants by Elkies and Atkin are known as efficient methods to count the orders of elliptic curves. Intelligent Choice System(ICS) was proposed to combine these methods efficiently. In this paper, we propose an improvement on the ICS. Further, we propose several implementation techniques in the characteristic 2 case.

1 Introduction

When we use the Elliptic Curve Cryptosystems [8,16] (*ECC* for short), we first have to define an elliptic curve over a finite field. Then, all cryptographic operations will be performed on the group of rational points on the curve and the security of ECC depends on the order of the group. (We call this order *the order of the curve* for short.) In order to avoid attacks that utilize some particular character of curves, such as MOV reduction [15] or SSSA attack [21,22,18], it is important to choose a curve randomly. Hence we need to count the order of a randomly selected curve defined over a finite field.

When the curve is defined over a large prime field, we can use the SEA (Schoof-Elkies-Atkin) algorithm [3,20,13] as the most efficient method to count the order of a curve. Thanks to Lercier [11] and Couveignes [10], we can also apply the SEA algorithm to curves over finite fields of characteristic 2. The SEA algorithm consists of two completely different stages of subprocedures and is a probabilistic algorithm. Thus, for an efficient implementation, it is important to find an efficient combination of such subprocedures. In [6] the Intelligent Choice System (*ICS* for short) was proposed to give an efficient combination of subprocedures in the first stage and was applied to counting the orders of curves over large prime fields.

In this paper, we propose an improvement on the ICS by considering not only the cost of the subprocedures in the first stage but also that of the second stage. Thus, the ICS is considered as a *practical optimization algorithm* which finds an efficient path through two different stages. Further, we introduce “two-cycles” subprocedure, by making use of the characters of the characteristic 2 fields. We also propose an implementation of [11] using Gröbner Basis.

The experiments show that we can count the order of an elliptic curve over $GF(2^{239})$ in 212 seconds in average on a Pentium II 400MHz PC. Incorporating with the *early abort* strategy [12], we can generate one secure curve over $GF(2^{160})$ within 266 seconds. Thus, we can generate secure elliptic curves quickly also in the characteristic 2 field case.

In Section 2, we briefly look over the SEA algorithm and the ICS. In Section 3, we propose an improvement on the ICS and apply it to the curves defined over finite fields of characteristic 2. In Section 4, we also propose a new subprocedure(two-cycles) and the implementation of [11] using Gröbner Basis. In section 5, we give our experimental results to show the new ICS works efficiently. We also give our results on generating secure curves for ECC. We note that a part of the results (Section 4) was already presented in [7] in Japanese.

2 SEA Algorithm and ICS

In this section, we recall the SEA algorithm and give brief overview of the ICS [6]. Here we consider an elliptic curve E over a finite field $GF(q)$ of q elements.

2.1 Schoof's Algorithm

First we will briefly recall the Schoof's algorithm [19]. We denote the subgroup of ℓ -torsion points of E by $E[\ell]$ and the Tate module by $T_\ell(E)$ for a prime ℓ . The *Frobenius endomorphism* $\phi : (x, y) \rightarrow (x^q, y^q)$ of E is defined on $T_\ell(E)$ as a linear map and satisfies the equation: $\phi^2 - t\phi + q = 0$, where t is the *trace* of ϕ which does not depend on ℓ . Then $\#E(GF(q)) = q+1-t$. If we find an integer t_ℓ such that

$$\phi^2(P) + qP = t_\ell\phi(P) \quad (1)$$

for any $P \in E[\ell]$, we get $t \equiv t_\ell \pmod{\ell}$. By Hasse's theorem, t must satisfy $-2\sqrt{q} \leq t \leq 2\sqrt{q}$. Therefore if we compute $t \pmod{\ell}$ for various small primes until their product exceeds $4\sqrt{q}$, we can uniquely determine the cardinality of the curve by means of the Chinese Remainder Theorem.

We denote the ℓ -th *division polynomial* by f_ℓ , whose degree is $(\ell^2-1)/2$ for $\ell \geq 3$, and which vanishes precisely on the x -coordinates of the ℓ -torsion points. As we compute (1) in the ring $GF(q)[x, y]/(e(x, y), f_\ell(x))$, where $e(x, y)$ is the defining polynomial of E , the dominant steps is the computation of x^q and y^q in that ring. From this, the complexity of this algorithm will be $O(\log^8(p))$.

2.2 SEA Algorithm and Isogeny Cycles

Elkies' idea is to make use of a degree $(\ell-1)/2$ factor g_ℓ of f_ℓ when it is possible to compute g_ℓ in $GF(q)[x]$. (In this case, ℓ is called an *Elkies prime*. Otherwise ℓ is called an *Atkin prime*). The factor g_ℓ represents an eigenspace of ϕ , which can be computed as a kernel of an isogeny map. Thus, $t \pmod{\ell}$ is calculated by the eigenvalue of the eigenspace. (We note that the ratio of Elkies primes is

expected as $1/2$.) This method will reduce the complexity to $O(\log^6(p))$. Rather than determining the unique value of $t \bmod \ell$, Atkin obtained certain restrictions on the value for Atkin prime case. Then the true value of t is found among a lot of candidates by the match-and-sort technique [10]. (See [20] for the detail.)

The SEA algorithm is obtained by combining the above two and consists of two stages, namely, (I) *collecting information stage* and (II) *trial search stage*:

Outline of the SEA algorithm

(I) Collecting informations on $t \bmod \ell$ for various ℓ 's until $\prod \ell > 4\sqrt{q}$:

- (1) Compute the modular polynomial $\Phi_\ell(x, j)$.
- (2) Check if $\bar{\Phi}(x) = \Phi_\ell(x, j(E))$ has a root in $GF(q)$.
 - (2-E) If $\bar{\Phi}(x)$ has a root in $GF(q)$ (we call ℓ an Elkies prime)
calculate $t_\ell = t \bmod \ell$ using g_ℓ .
 - (2-A) Otherwise (we call ℓ an Atkin prime)
calculate possible values of $t \bmod \ell$. We denote the set of possible values of $t \bmod \ell$ by T_ℓ .

We denote the set of Elkies primes and the set of Atkin primes by \mathcal{E} and \mathcal{A} , respectively.

(II) Determining the value t by trial search: Now, there are a lot of candidates T for the value of t , where

$$T \bmod \ell = t_\ell \text{ for } \ell \in \mathcal{E} \text{ and } T \bmod \ell \in T_\ell \text{ for } \ell \in \mathcal{A}.$$

The value of t is (uniquely) determined by testing if $(q + 1 - T)P = \mathcal{O}$ for each candidate T , where P is a sample rational point of E and \mathcal{O} is the point of infinity.

The isogeny cycles method [3] was proposed to improve the SEA algorithm, where $t \bmod \ell^2, t \bmod \ell^3, \dots$ can be computed efficiently from $t \bmod \ell$ when ℓ is an Elkies prime. Incorporating the *isogeny cycles* method, the stage (I) allows information on $t \bmod \ell^k$ for some positive integer k . Thus, in the stage (I), we will gather informations of $t \bmod \ell^k$ for some primes ℓ and integers $k > 0$. We call the product of all primes or prime powers the **counter** whose information will be used in the stage (II). When **counter** exceeds $4\sqrt{q}$ in the stage (I), we enter the next stage (II).

2.3 ICS

In implementation of SEA, the following choices are very important for the total efficiency;

1. decision whether to apply the isogeny cycles method for $t \bmod \ell^k$ or not, when we find an Elkies prime ℓ ,
2. decision whether to compute the candidates for the value of t or just abandon it, when we find an Atkin prime, and
3. the setting on the upper bound **CanMAX** on the number of candidates of the trace and usage of informations with respect to Atkin primes.

Thus, to give the most efficient realization (implementation), we have to find good combinations (choices) of subprocedures. In [6], this “combination problem” was dealt with and the ICS was proposed as a systematic and practical answer. It has the following functions corresponding to the problem.

The functions of the ICS [6]

- (A) At each step in the stage (I), the subprocedure M with the smallest *estimated complexity* $\text{complex}(M)$ is chosen among the possible subprocedures. Usually, $\text{complex}(M)$ is set by the complexity of the dominant steps of M .
- (B) For each Atkin prime ℓ , the decision if we use the information on $t \bmod \ell$ in the stage (II) will be made by introducing an index (*Atkin index*).
- (C) By the *virtual (Atkin/isogeny cycles)* methods, we can enter the stage (II) from the stage (I) *dynamically*.

When we handle curves over large prime fields $GF(p)$, the ICS chooses one from the following at each step in the stage (I): (See Figure 1.)

- (a) Schoof's original algorithm,
- (b) Atkin/Elkies' method,
- (c) the isogeny cycles method,
- (d) the virtual method (a short cut to the stage (II)) proposed in [6].

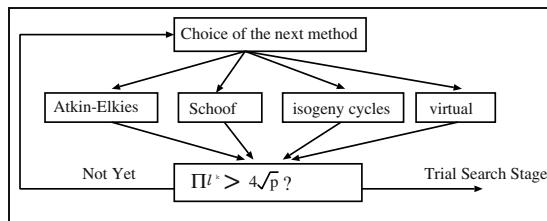


Fig. 1. The diagram of the ICS

3 The New ICS for Curves over $GF(2^n)$

In order to apply the ICS to the characteristic 2 case efficiently, we enhance the ICS as follows:

- (i) We find an efficient path from the stage (I) to the stage (II).
- (ii) We add a special subprocedure which makes a good use of the speciality of the characteristic 2.

We note that the enhancement (i) is also applicable to the order counting of curves over large prime fields.

3.1 Finding an Efficient Path from Stage (I) to Stage (II)

In the original ICS we set the upper-bound `CanMAX` on the total number of candidates of t which will be tested in the stage (II). However, to get the best total timings, we have to take care of the cost of the stage (II) and find an efficient path (flow of computations) from the stage (I) to the stage (II).

To find such an efficient path, we consider the virtual method as a procedure between the stage (I) and the stage (II). That is, one can decide to enter the stage (II) even if `counter` does not exceed the bound $4\sqrt{q}$. In this case, we apply the virtual method before the computation in the stage (II).

Stage (I) → virtual method if necessary → Stage (II)

At each step in the stage (I), the choice of the next subprocedure and the decision whether one enters the stage (II) or not are made at the same time by the new function `Single-Depth-Search` (`SDS` in short). Here, we introduce a subprocedure M_0 which does nothing. We call M_0 *no-operation*. Let other possible subprocedures be M_1, \dots, M_k .

Single-Depth-Search:

- (1) Compute the estimate $\text{ET}(M)$ of total time for each subprocedure M as follows:
 - (i) For M_0 , estimate the total time when one enters the stage (II) directly. (Between the stage (I) and stage (II), one executes the virtual method if necessary.)
 - (ii) For other M_i , $1 \leq i \leq k$, estimate the total time when one executes M_i and then enters the stage (II). (Between the stage (I) and stage (II), one executes the virtual method if necessary.)
- (2) Among $\text{ET}(M_0), \dots, \text{ET}(M_k)$, find the smallest one, say $\text{ET}(M_s)$. If $M_s = M_0$, one enters the stage (II). Otherwise, one chooses M_s as the next procedure.

Repeating the function `SDS` at each step, we have a *practical realization* of `multiple-depth-search` which will give the most efficient flow of computations. The new ICS has the following functions: (See Figure 2.)

The functions of the new ICS with SDS

- (N-A) At each step in the stage (I), the subprocedure with the smallest *estimated total time* is chosen among the possible subprocedures by the function `SDS`. If the output is M_0 , we enter the stage (II).
- (N-B) For each Atkin prime ℓ , the decision if we use the information on $t \bmod \ell$ in the stage (II) will be made by introducing an index (Atkin index).

Next, we will give the details of estimating the total time.

Estimate of the Total Time: Let M be the chosen method and N_i the expected number of the candidates in the stage (II) after the execution of M with probability p_i for each i . Then, the estimate $\text{ET}(M)$ of the total time is given by

$$c_M \text{complex}(M) + \sum_i c_{m-s,i} p_i \sqrt{N_i}, \quad (2)$$

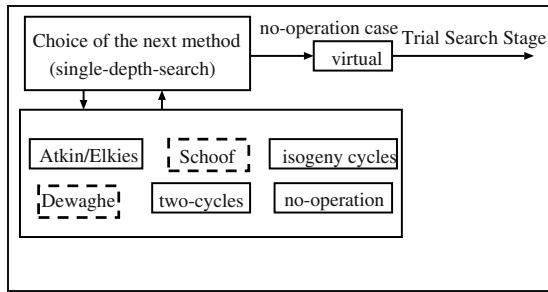


Fig. 2. The diagram of the new ICS

where $c_M, c_{m-s,i}$ are *weights* (in \mathbf{R}) and $\text{complex}(T)$ denotes the complexity of the dominant step of T over the field arithmetics, which is also used in the function (A) in the original ICS. We determine the weights $c_M, c_{m-s,i}$ by data of actual computations on a real computer. Of course, these depend on the computational environment (the CPU, the compiler, *etc.*). We note that, for the Atkin/Elkies's method for a prime ℓ , the expected information on t is $t \bmod \ell$ with 1/2 possibility and a set \mathcal{T}_ℓ of candidates with 1/2 possibility. From these, we estimate the expected number of candidates.

3.2 Additional Subprocedures of ICS for Curves over $GF(2^n)$

We can use the following special subprocedures (e),(f) in the stage (I) in the SEA algorithm, by which we can improve the effect of the ICS. (For the large prime field case, the method (f) is also applicable, but the method (e) is not, since the method (e) uses the special characters of characteristic 2 fields.)

- (e) A method to compute $t \bmod 2^m$ for positive integers m ,
- (f) A method based on Dewaghe's idea [4] to compute $t \bmod \ell$ for an Atkin prime ℓ .

For simplicity, we call the method (e) the *two-cycles method*, and the method (f) *Dewaghe's method*. Compared with the methods (b) and (c) in the original ICS in Section 2.3, the methods (e) and (f) do not seem to have much "contribution" in the stage (I), however, they seem to work very well as "short cuts" to the stage (II). We will give details of two-cycles method in Section 4.1 and show its practical effect in Section 5.1. As our current implementation does not support Dewaghe's method, we omit its description.

Remark 1 In the author's experiments, we computed $t \bmod 2^k$ for appropriate k at the beginning of the SEA algorithm.

3.3 Effects of the new ICS in Characteristic 2 Case

The original ICS requires the upper-bound `CanMAX` on the total number of candidates of t , on which the total efficiency depends heavily. To get an appropriate

CanMAX, we have to measure the timings on a number of examples. On the other hand, in the new ICS, the effect of **CanMAX** on the total efficiency becomes much smaller and we can find more efficient value of **CanMAX**. In the original ICS, **CanMAX** for curves over $GF(q)$ with 155 bit size q was set as 10^8 , but it can be increased to 10^9 in the new ICS. Of course, the function **SDS** has actual effect in the case where the number of candidates becomes close to **CanMAX**.

Examples 2 *Here we demonstrate the detail of actual computation by examples on a PC with Pentium II 400MHz CPU. We select examples for which the function **SDS** works well. In the below, series of triples represent the trace of computation, where each triple consists of the selected method, the selected prime, and the actual cost (in seconds). For simplicity, we write **a,e** and **i** for Atkin's method, Elkies' method and the isogeny cycles method, respectively. (For the definition of E_β , see Section 4.1.)*

(1) E_β , where $\beta = \alpha^5 + \alpha^3$, over $GF(2^{155}) = GF(2)(\alpha)$. It took 27.5 seconds by the ICS without **SDS**, however, it took 18.9 second by the ICS with **SDS**. We list the final steps in the computations.

no **SDS**: $[e, 41, 2.6], [a, 43, 2.3] \rightarrow$ virtual method \rightarrow Stage (II)

SDS: $[e, 41, 2.6], [a, 43, 2.3], [e, 47, 3.1] \rightarrow$ virtual method \rightarrow Stage (II)

(2) E_β , where $\beta = \alpha^6 + \alpha^5 + \alpha^2 + 1$, over $GF(2^{196}) = GF(2)(\alpha)$. It took 119 seconds by the ICS without **SDS**, however, it took 95.7 seconds by the ICS with **SDS**. We list the final steps in the computations below.

no **SDS**: $[a, 67, 11.0], [e, 71, 14.1], [a, 73, 14.6] \rightarrow$ virtual method \rightarrow Stage (II)

SDS: $[a, 67, 10.9], [e, 71, 14.1], [i, 11, 6.8] \rightarrow$ virtual method \rightarrow Stage (II)

4 Implementations of Subprocedures

In order to make the ICS work efficient, subprocedures must be efficiently computable. Here we pick up subprocedures which are important and completely different from subprocedures for odd characteristic fields, and give some details on them. (Cf. [7].) From now on, let $K = GF(2^n)$.

4.1 Two-Cycles Method for Curves over $GF(2^n)$

The two-cycles method utilizes the special characters of the characteristic 2 fields. Here, we consider elliptic curves defined as follows:

$$E_a : y^2 + yx = x^3 + a, \quad a \in K \tag{3}$$

We note that each elliptic curve defined over K is K -isomorphic to some E_a or its quadratic twist.

The two-cycles method uses the fact that $E_a[2^k] \cong \mathbf{Z}/2^k\mathbf{Z}$ is the eigenspace of the Frobenius endomorphism and the 2^k -division polynomial can be computed efficiently. Let $g_0 = 0$, $g_1 = x + \sqrt[4]{a}$, and for each $i > 1$

$$g_i = g_{i-1}^2 + \sqrt[2^{i+1}]{a} x \prod_{j=1}^{i-2} g_j^2. \tag{4}$$

Then, for points $P = (x, y), Q = (x', y')$ on E with $Q = 2^k P$, we have

$$x' = \frac{g_k^{2^{k+1}}}{h_k^{2^k}}, \quad (5)$$

where $h_k = x \prod_{j=1}^{k-1} g_j^2$. (See [14].) Since the point of order 2 on E_α is $(0, \sqrt{\alpha})$, the x -coordinate of each point of order 2^k is a root of $g_{k-1}(x)$. By generalizing this fact, we have the following. (We note that the x -coordinate of a point P of order 2^k is K -rational, then P is K -rational.)

Lemma 3 Suppose that a point of order 2^k is K -rational and its x -coordinate, say α , is already computed. Then, the x -coordinate of each point of order 2^{k+s} is a root of the following polynomial of degree 2^s .

$$g_{k,s} = g_s^2 + \sqrt[2^s]{\alpha} h_s = 0 \quad (6)$$

Each point of order 4 is K -rational and so $t \equiv 1 \pmod{4}$. Then, the x -coordinate of each point of order 8 is a root of the polynomial g_2 of degree 2. Factoring g_2 , we get the x -coordinate of a point of order 8, if g_2 has a rational root, or we get $t \equiv 5 \pmod{8}$, otherwise. Repeating this procedure, we can compute $t \equiv 1 \pmod{2^k}$ efficiently if each point of order 2^k is K -rational.

Examples 4 Let $K = GF(2^{155}) = GF(2)(\alpha)$, where $\alpha^{155} + \alpha^{62} + 1 = 0$. On the curve $E_\alpha : y^2 + yx = x^3 + \alpha$, points of order 2^{20} are K -rational. In this case, the two-cycles method works efficiently and the order $\#E_\alpha(K)$ was computed within 7.6 seconds on a PC with Pentium II 400MHz CPU.

When we compute $t \pmod{2^{k+s}}$ by using $g_{k,s}$, we can use a technique similar to the match-and-sort technique in the isogeny cycles method ([6]). Since the value $t' = t \pmod{2^{k+s-1}}$ is already computed, either $t \equiv t' \pmod{2^{k+s}}$ or $t \equiv t' + 2^{k+s-1} \pmod{2^{k+s}}$ holds. Thus, finding a pair (u, v) of integers such that (i) either $ut' = v$ or $u(t' + 2^{k+s-1}) = v$ holds, and (ii) $u^2 + v^2$ is as small as possible, and testing if

$$u\phi(P) = \pm vP, \quad (7)$$

we can determine the value $t \pmod{2^{k+s}}$.

Estimate of Complexity: In the new ICS, we can use the following estimate on the complexity. Suppose that we have already computed $t \pmod{2^{k+s-1}}$. Then we consider that the computation of $t \pmod{2^{k+s}}$, requires

$$(n + c \max\{|u|, |v|\}) \times U^2, \quad (8)$$

where $U = \deg(g_{k,s})$, n is the extension degree of K and c is a constant.

4.2 Isogeny Map Computation by Lercier's Method

Here, we will discuss on the efficiency of the isogeny map computation for curves over $GF(2^n)$. In the implementation, we tested Lercier's method [11], Couveignes's second method [2] and a simple modification of Couveignes's second method based on interpolation and we found that Lercier's method is more efficient than others, even in the case suited for Couveignes's second method, i.e. a given curve has rational points of a large order 2^k . To give a practical implementation of Lercier's method, we utilize Gröbner basis computation facility of Risa/Asir computer algebra system [17]. Then, the ratio of the computation of isogeny maps in the whole computation is quite small. For example, the ratio is around 7 % for curves over $GF(2^{155})$ and $GF(2^{196})$.

Implementation with Gröbner Basis Computation In Lercier's method [11], one computes the isogeny map by solving a certain system \mathcal{S} of algebraic equations over K and those algebraic equations are derived from the commutativity of the isogeny map and the translation of the point of order 2. In a practical point of view, the key of Lercier's method is that \mathcal{S} has the unique solution belonging to $GF(2)^k$, where k is the number of indeterminates in \mathcal{S} . Since \mathcal{S} is defined over K , we can transform \mathcal{S} to another system \mathcal{S}_0 of algebraic equations over $GF(2)$ with the same indeterminates. Then \mathcal{S}_0 becomes so-called an “over-determined” system. To solve such a system, the Gröbner basis computation seems very useful and efficient. (See [1] for the notion and algorithms of Gröbner basis.)

Now, suppose that π_1, \dots, π_k are indeterminates in the system \mathcal{S}_0 and $I(\mathcal{S}_0)$ is the ideal in $GF(2)[\pi_1, \dots, \pi_k]$ generated by all polynomials in \mathcal{S}_0 . Since \mathcal{S}_0 has the unique solution, with respect to any term order, the Gröbner basis of $I(\mathcal{S}_0)$ is $(\pi_1 - e_1, \dots, \pi_k - e_k)$, where $e_i \in GF(2)$ and (e_1, \dots, e_k) is the solution of \mathcal{S}_0 .

By several experiments, we observe that many of equations in \mathcal{S}_0 are linear. Moreover, there seems a certain randomness in the distribution of coefficients of equations, which implies that for each π_i , the ratio of equations having a term containing π_1 is close to 1/2. From these observation and the property of the “sugar” strategy [5] in the computation of the Gröbner basis, the computation of the Gröbner basis with sugar strategy becomes close to the computation of solving of a system of sparse linear equations. Hence, the Gröbner basis can be computed quite efficiently in this case. As a result, the cost of generating algebraic equations is much larger than that of solving the system in the experiments.

Of course, the Gröbner basis computation is a general method which can handle non-linear equations. So, it can also compute “exceptional” cases, where the system \mathcal{S}_0 is not sparse or has a large degree, precisely.

5 Experimental Results

We have implemented the new ICS using Risa/Asir computer algebra system [17] and examined its ability and efficiency by experiments on a number of examples

on a PC with Pentium II 400 MHz CPU. We used the same settings on the complexity estimate function `complex` for Atkin/Elkies methods and the isogeny cycles method as in [6], and made a special table for the two-cycles method based on data of actual computations. Same as in [6], we did not use the original Schoof's method. For the experiment, we pre-computed modular polynomials for curves over $GF(2^n)$. The timings of the experiment and the detailed analysis on the timings certainly show the efficiency/progress of the new ICS and the implementations of subprocedures.

5.1 Order Counting

We selected four fields $GF(2^{155})$, $GF(2^{160})$, $GF(2^{196})$, $GF(2^{239})$, chose 200 curves over each field as

$$y^2 + yx = x^3 + a, \quad 2 \leq n(a) \leq 201,$$

and measured the average time needed to compute the order of one curve. (For $GF(2^{239})$, we chose 150 curves.) Here, $n(a)$ denotes the binary expression of an element a in K , that is, $n(a) = \sum a_i 2^i$ if $a = \sum a_i \alpha^i$, where $a_i \in \{0, 1\}$ and α is the primitive element. Moreover, in order to analyze the effect of the ICS, we tried several combinations of functions of the ICS and applied them for 200 curves over $GF(2^{160})$.

Table 1 shows the timings of counting the orders of curves over each field. We also put the best and the worst time in the table. To know the effect of the function `SDS`, we tested the ICS with `SDS` and that without `SDS`. The speed up (the progress) by the `SDS` is almost 3 % and the expected “practical complexity” seems around $O(n^5)$ which also shows the quality of the implementation. In smaller field, as the two-cycles method works very well, the effect of the `SDS` is rather small. Table 2 shows the detailed analysis on the effect of each function in the ICS, where `SDS` means the use of `SDS`, `re-ordering` means the use of Atkin index (see Section 2.3) and `isogeny` means the use of `complex` to choose the isogeny method. So, the number (3) corresponds to the original ICS applied to curves over $GF(2^n)$ and the number (8) corresponds to the ordinary SEA algorithm with isogeny cycles method. Table 2 suggests that the new ICS attained nearly twice faster than the ordinary SEA with isogeny cycles method. Moreover, it also suggests that our further enhancement on the ICS improved the efficiency nearly 40 % up compared with the original ICS.

5.2 Finding Secure Elliptic Curves

For secure ECC, it is strongly recommended to use a curve whose order is prime or almost prime. For this purpose, we can use *early abort* strategy [12]. In this strategy, we check if the order has a factor in each step of the computation of $t \bmod \ell$. If we find that the order is not almost prime, we can abandon the curve and try the next one. The effect of the strategy is supported by mathematical

Table 1. Timings for counting orders (seconds):

degree	SDS	average time	best time	worst time	CanMAX
155	USE	20.3	7.6	42.7	10^9
	NO	21.0	8.6	44.5	10^8
160	USE	22.2	14.2	50.3	10^9
	NO	22.8	15.5	50.2	10^8
196	USE	79.8	40.4	155.4	10^{10}
	NO	82.2	40.7	149.7	5×10^8
239	USE	212.0	107.4	578.2	5×10^{10}
	NO	216.0	113.7	548.7	10^9

Table 2. Effects of functions of the ICS (seconds):

No.	SDS	two-cycle	isogeny	re-ordering	virtual	average	best	worst
(1)	YES	YES	YES	YES	YES	22.2	14.2	50.3
(2)	NO	YES	YES	YES	YES	22.8	15.5	50.2
(3)	NO	NO	YES	YES	YES	31.0	19.1	77.5
(4)	NO	YES	NO	YES	YES	25.9	16.6	53.3
(5)	NO	YES	YES	NO	YES	27.1	15.2	88.8
(6)	NO	YES	YES	YES	NO	29.6	20.1	50.5
(7)	NO	NO	YES	YES	NO	38.1	23.3	77.5
(8)	NO	NO	NO	NO	NO	42.6	37.3	78.4

analysis [9,12]. Here, we incorporated the strategy to our implementation with the ICS and searched a number of secure curves.

For $GF(2^{160})$ and $GF(2^{239})$, we generated curves E_a randomly and tested if the order of the quadratic twist of E_a is written as $2 \times p$ for some prime number p . So, we searched “secure curves” among the quadratic twists of randomly generated curves. We generated 20 “secure” curves over each field and list the average time to find one curve in Table 3. From the timings, we are convinced that one can generate secure curves quite efficiently.

Table 3. Secure Curve Generation (seconds):

finite field	$GF(2^{160})$	$GF(2^{239})$
average time	266	3783

Remark 5 We note that even if the order $\#E_a$ is divisible by larger power of 2, its quadratic twist is not divisible by 4. Thus, the two-cycles method never does harm in finding a secure curve.

6 Concluding Remarks

We have introduced the new ICS and could speed up the order counting process 40 %. Comparing with the process without the ICS, the new ICS process is more than 90 % faster. We also proposed several practical implementations that can be used in the characteristic 2 case. As a result, incorporating with the early-abort strategy, we can choose secure curves from randomly selected elliptic curves in a short time. For example, our experiment shows that we can generate one secure curve over $GF(2^{160})$ within 266 seconds. We note that for order counting of curves over large prime fields our current implementation with SDS is about 20 % faster for curves over $GF(2^{240} + 115)$ than that in [6], where we do not use fast arithmetics technique.

Our future work will be speeding up the process and calculation of curves defined over bigger fields. The important factors are (1) finding the better settings in the ICS, (2) improvements on the match-and-sort algorithm, (3) improvements on isogeny calculation, and (4) speeding up the calculation of eigenvalues.

The new ICS can handle any efficient subprocedures and it can be applied to any elliptic curves over any ground fields (e.g. OEF [23]) in the SEA algorithm.

References

1. Becker, T., Weispfenning, V., Gröbner Bases. Graduate Texts in Mathematics 141, Springer-Verlag (1993). [218](#)
2. Couveignes, J-M., Computing ℓ -isogenies using the p -torsion, *ANT-II*, Lecture Notes in Computer Science, 1122, Springer (1996) 59–65. [218](#)
3. Couveignes, J.-M., Dewaghe, L., Morain, F., Isogeny cycles and the Schoof-Elkies-Atkin algorithm, LIX/RR/96/03 (1996). [210](#), [212](#)
4. Dewaghe, L., Remarks on the Schoof-Elkies-Atkin algorithm, *Mathematics of Computation* **67** (1998) 1247–1252. [215](#)
5. Giovini, A., Mora, T., Niesi, G., Robbiano, L., Traverso, C., One sugar cube, please, *Proceedings of ISSAC '91*, ACM Press (1991) 49–54. [218](#)
6. Izu, T., Kogure, J., Noro, M., Yokoyama, K., Efficient implementation of Schoof's algorithm, *ASIACRYPT'98*, Lecture Notes in Computer Science, 1514, Springer (1998) 66–79. [210](#), [211](#), [213](#), [217](#), [219](#), [221](#)
7. Izu, T., Kogure, J., Noro, M., Yokoyama, K., Order counting of elliptic curves defined over finite fields of characteristic 2, *Transaction of IEICE Series A*, **82** (1999) 1253–1260. (in Japanese) [211](#), [216](#)
8. Koblitz, N., Elliptic curve cryptosystems, *Mathematics of Computation* **48** (1987) 203–209. [210](#)
9. Lenstra Jr., H.W., Factoring integers with elliptic curves, *Annals of Mathematics* **126** (1987) 649–673. [220](#)
10. Lercier, R., Algorithmique des courbes elliptiques dans les corps finis, Doctoral Thesis, L'École Polytechnique (1997). [210](#), [212](#)
11. Lercier, R., Computing isogenies in F_{2^n} , *ANTS-II*, Lecture Notes in Computer Science, 1122, Springer (1996) 197–212. [210](#), [211](#), [218](#)
12. Lercier, R., Finding good random elliptic curves for cryptosystems defined over F_{2^n} , *EUROCRYPT '97*, Lecture Notes in Computer Science, 1233, Springer (1997) 379–392. [211](#), [219](#), [220](#)

13. Lercier, R., Morain, F., Counting the number of points on elliptic curves over finite fields: strategy and performances, *EUROCRYPT '95*, Lecture Notes in Computer Science, 921, Springer (1995) 79–94. [210](#)
14. Menezes, A., Elliptic curve public key cryptosystems, Kluwer Academic Publishers, Boston (1993). [217](#)
15. Menezes, A., Okamoto, T., Vanstone, S.E., Reducing elliptic curves logarithms to logarithms in a finite field, *STOC '91*, ACM Press (1991) 80–89. [210](#)
16. Miller, V., Uses of elliptic curves in cryptography, In *CRYPTO '85*, Lecture Notes in Computer Science, 218 (1986) 417–426. [210](#)
17. Noro, M., Takeshima, T., Risa/Asir – a computer algebra system, *Proceedings of ISSAC '92*, ACM Press, New York (1992) 387–396. [218](#)
18. Satoh, T., Araki, K., Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves, *Commentarii Math. Univ. Sancti Pauli* **47** (1997) 865–874. [210](#)
19. Schoof, R., Elliptic curves over finite fields and the computation of square roots mod p, *Mathematics of Computation* **44** (1985) 483–494. [211](#)
20. Schoof, R., Counting points on elliptic curves over finite fields, *J. Théor. Nombres Bordeaux* **7** (1995) 219–254. [210, 212](#)
21. Semaev, I.A., Evaluation of discrete logarithms in a group of p -torsion points of an elliptic curves in characteristic p , *Mathematics of Computation* **67** (1998) 353–356. [210](#)
22. Smart, N.P. The discrete logarithm problem on elliptic curves of trace one, preprint (1997). (to appear in *J. Cryptology*) [210](#)
23. Vailey, D.V., Paar, C., Optimal extension fields for fast arithmetic in public-key algorithms, *CRYPTO'98*, Lecture Notes in Computer Science, 1462, Springer (1998) 472–485. [221](#)

Key Recovery in Third Generation Wireless Communication Systems^{*}

Juanma González Nieto¹, DongGook Park^{1,2}, Colin Boyd¹, and Ed Dawson¹

¹ Queensland University of Technology, Information Security Research Centre,
GPO Box 2434, Brisbane, Queensland 4001, Australia

² Korea Telecom, Access Network Laboratory,
17 WooMyeon-Dong, SeoCho-Gu, 137-792, Seoul, Korea

Abstract. Future mobile communications networks, so called third generation systems, may need end-to-end security in some applications involving value-added services such as providing secure communications between a user and a bank in electronic commerce. The provision of end-to-end security may require mechanisms for key recovery. In this paper we identify security flaws with a previous published protocol for key recovery in such networks. A new key recovery protocol which overcomes these flaws is presented.

1 Introduction

Third generation wireless mobile communication systems will realise a new vision of telecommunication services through the provision of enhanced current generation services such as cellular, PCS and radio-paging within their unified architectures, and of a more diverse range of telecommunication services including multimedia and Internet services. Two main systems have been proposed for future generation mobile applications: the International Mobile Telecommunications-2000 (IMT-2000) [16,6] on a worldwide basis, and the Universal Mobile Telecommunications System (UMTS) [4] in the European context. UMTS, like its predecessor Global System for Mobile (GSM), seems to have influenced worldwide standardisation development for third generation systems.

UMTS and IMT-2000 will take advantage of many advanced security technologies, especially asymmetric (or public key) cryptography, at least in their fully developed stages [15]. Public key based security, with a mature public key infrastructure (PKI), will enable all the involved entities such as users, network operators (NOs), value-added service providers (VASPs) and service providers (SPs) to have full range of state-of-the art security features including:

- non-repudiation services for incontestable charging and electronic commerce.
- mutual authentication between the user and the NO/VASP without accessing the home SP on-line, thus enabling seamless roaming of the user.

* This research is part of the co-operative project Security Technologies in Wireless Communications between Queensland University of Technology and Korea Telecom

All the security features depend on the successful execution of a wireless authentication and key establishment (WAKE) protocol between the user and the NO/VASP. Figure 1 shows all the entities involved in the public key based security architecture of future mobile communications as well as the interfaces over which WAKE protocols will be run.

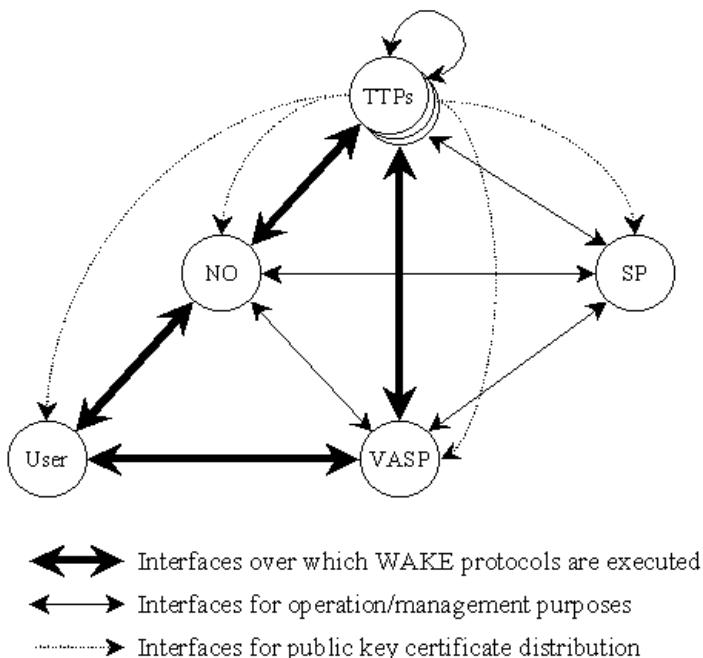


Fig. 1. Role players in future wireless mobile communications using public key infrastructure

Trusted third parties (TTPs) will serve as clearing houses with regard to all the required trust relations for WAKE protocol executions between the user and the NO/VASP, both on an on-line and off-line basis. The User-to-VASP interface is a logical interface to be established through NO networks, and has some interesting issues such as a rather strict requirement for non-repudiation services for electronic payment, and some appropriate key recovery mechanisms. The latter is the main issue of this paper.

The user and the VASP need to authenticate each other and establish a shared secret session key to encrypt the subsequent message exchange. This is achieved through execution of the relevant WAKE protocol which may or may not be the same as the WAKE protocol for the user-to-NO interface. Because VASPs, unlike NOs and SPs, are a special kind of end user, the user-to-VASP

communication can be regarded an end-to-end link, which gives rise to a problem in relation to lawful interception. The information protected with user-to-NO session keys can be accessed by law enforcement agencies (LEAs) with the aid of the relevant NO (when authorized) without using any special cryptographic mechanisms. End-to-end encryption, however, requires a special cryptographic facility, a *key recovery* mechanism, to enable a LEA to access the plaintext encrypted under an end-to-end encryption key. Key recovery, although still a matter of debate, is expected to be deployed in the UMTS security architecture to satisfy the government requirement of lawful interception [12].

In this paper, we first review a proposed key recovery protocol for the prominent UMTS WAKE protocol. This is followed by a security analysis of this protocol which identifies several vulnerabilities. A new key recovery protocol is then proposed which is not only secure from the vulnerabilities of the UMTS proposal but is also more computationally efficient. The proposed new protocol utilises a simple mechanism for verifiable encryption appropriate for use in this context.

2 Background on Key Recovery

Cryptography protects the confidentiality of information by limiting access to plaintext of encrypted data to those that possess the corresponding decryption keys. This in turn requires the deployment of key management techniques for the secure administration (generation, distribution, storage, etc) of cryptographic keys. In particular, mechanisms might be needed to allow extraordinary access to the plaintext data by authorised parties in cases where the corresponding decryption keys are not otherwise available [8]. This usually involves a Trusted Third Party (TTP) that has the capability of restoring the appropriate decryption keys and this process is generically called *key recovery* (KR). Two typical scenarios where KR may be needed are:

- when the decryption key has been lost or the user is not present to provide the key.
- where commercial organisations want to monitor their encrypted traffic without alerting the communicating parties, for example to check that employees are not violating the organisation's policies.

National governments have also shown interest in the deployment of key recovery techniques, mainly motivated by law enforcement and intelligence concerns about the reduction in their capability for wiretapping when strong cryptography is used. This concern has led to proposals such as the famous Clipper Chip in the USA and the GCHQ protocol in the UK [USD94, CES96].

In this paper we use the following terminology:

Key Recovery Agent (KRA) Trusted third party that performs KR in response to an authorised request.

Key Recovery Information (KRI) Aggregate of data that is needed by the KRA in order to complete a KR request, e.g. a session key encrypted under the KRA's public key.

Key Recovery Requester (KRR) Authorised entity that requests KR from the KRA. The KRR would usually be a LEA in possession of a valid warrant.

Interception Agent Entity that acts in response to an authorised request for interception on a target identity by filtering out the communications traffic corresponding to such target identity. This function would usually be performed by NOs [13,11].

Rantos and Mitchell [17] proposed a KR scheme for UMTS as part of the European Community research project on Advanced Security for Personal Communications Technology (ASPeCT) [2]. The authors' strategy was to modify the already designed and well-studied ASPeCT WAKE protocol for the user-to-VASP interface [14]. These authors identified the following goals and requirements for the KR enhanced version of the WAKE protocol:

- R1 Recoverability of the session key K by the corresponding KRA. This is clearly the main goal: at the end of a successful run of the protocol the KRA that each entity is associated with should be capable of restoring K when presented with the available KRI.
- R2 Minimal computational overhead. An overriding constraint in mobile communications is the limited computational power of the mobile equipment since usually cryptographic operations will be performed on smart cards. Rantos and Mitchell also state that the computational overhead at the user end should be kept at the same level.
- R3 Unobtrusiveness. The enhanced protocol should not introduce any vulnerability into the ASPeCT protocol. This would occur if any of the security services provided originally by the protocol was compromised by the added KR mechanism. Obviously we have to make an exception to this requirement, with regard to the confidentiality of the communications which, in the KR enhanced version, can be revoked by the corresponding KRAs.
- R4 Fine granularity. The number of session keys that can be recovered from a single instance of KRI should be small enough so as to ensure fine granularity of authorised interception periods. In other words, KR that have been authorised for a specified period should not compromise communications outside the scope of the authorisation.

In the following two sections we describe and analyse the properties of the KR enhanced ASPeCT protocol. It turns out that at least one of the above requirements is actually not met by the protocol. In particular the unobtrusiveness requirement is not achieved, which allows an impersonation attack in the protocol. We also propose a fix to prevent such an attack as well as a new KR mechanism which exhibits additional and improved properties.

3 Key Recovery for ASPeCT Protocol

3.1 ASPeCT Protocol

The most well known public key based WAKE protocol for UMTS is proposed by the ASPeCT project, which is responsible for the research and development of security technologies to be used in the UMTS system. We shall call this protocol the ASPeCT protocol in this paper. The ASPeCT protocol is proposed for both user-to-network and user-to-VASP interfaces to enable economical deployment of security services. Detailed descriptions of the ASPeCT protocol can be found in the literature [14,1]. The message flows are shown in figure 2, where all the charging related data fields are omitted for simplicity. The notation used in this, and subsequent protocol descriptions, is shown in table 1.

A	the identity of the user
B	the identity of the VASP
TTP_A	the identity of the TTP of user A
g	a generator of a finite group
r_A	a random nonce chosen by user A
r_B	a random nonce chosen by the VASP, B
K_{AB}	a secret session key established between the user and the VASP, B
A_{Cert}	public key certificate of the user, A
B_{Cert}	public key certificate of the VASP, B
b	the private key component of the public-private key-agreement key pair of the VASP, B
g^b	the public key component of the public-private key-agreement key pair of the VASP, B
$\{m\}_{K_A^{-1}}$	the message m signed by the user with his/her private signature key K_A^{-1}
$\{m\}_{K_{AB}}$	the symmetric encryption of a message m using the session key K_{AB}
h_1, h_2, h_3	one-way hash functions

Table 1. Notation for Protocol Descriptions

A: user, B: VASP, TTP_A : TTP of A

1. $A \rightarrow B : g^{r_A}, TTP_A$
2. $A \leftarrow B : r_B, h_2(K_{AB}, r_B, B), B_{Cert}$
3. $A \rightarrow B : \{\{h_3(g^{r_A}, g^b, r_B, B)\}_{K_A^{-1}}, A_{Cert}\}_{K_{AB}}$

$A, B : K_{AB} = h_1(r_B, g^{br_A})$

Fig. 2. The ASPeCT protocol

According to Horn and Preneel [14], the protocol satisfies the following six goals:

1. mutual explicit authentication of A and B ;
2. agreement between A and B on a secret session key K_{AB} with mutual implicit key authentication;
3. mutual key confirmation between A and B ;
4. mutual assurance of key freshness (mutual key control);
5. non-repudiation of origin by A for relevant data sent from A to B ;
6. confidentiality of relevant data sent by A to B .

The ASPeCT protocol is related to the Station-to-Station (STS) protocol of Diffie *et al.* [10] in that both protocols use the same challenge-response mechanism, i.e., A and B challenge each other with random nonces (g^{r_A} and r_B in this protocol) exchanged in clear and calculate responses using private keys (K_A^{-1} and b respectively in this protocol). In this protocol, however, it should be noted that the second message in the protocol does not contain any signature by B . The third message includes the signature by A like STS protocol to accommodate non-repudiation requirement for the relevant message from the user. In fact, the protocol shown in figure 2 is one of two variants of the ASPeCT protocol for user-VASP application, which is called variant B. Another variant, so called variant C, is an extended version of the protocol to include on-line TTPs, where the first message from A to B include the user identity encrypted under the key $L = (g^{x_A})^{r_A}$, where g^{x_A} is the public key agreement key of KRA_A . This new field allows the TTP of user A to identify the user, verify whether the user's certificate has been revoked, and deliver the user's certificate A_{Cert} to the VASP.

3.2 KR Enhanced ASPeCT Protocol

In the KR enhanced ASPeCT protocol proposed by Rantos and Mitchel [17], each entity A and B , registers with a KRA, KRA_A and KRA_B , in their respective domains. The same TTP is assumed to act both as the certification authority (CA) and the KRA for each entity. Two different solutions for KR are proposed that can be applied to both variants of the ASPeCT protocol. For brevity, we only describe the B-variant protocol. The extrapolation to the C-variant protocol is straightforward. Figure 3 illustrates the first of the two given solutions.

The KR capability is achieved by modifying the way in which r_A is generated. Specifically, r_A is now computed as

$$r_A = f(w_A, s_A)$$

where f is a one-way function, s_A is a one-time random seed, and w_A is a secret value shared between A and KRA_A which has been previously established between the two of them during the registration phase. Message 1 also includes A 's identity encrypted under the key $L = (g^{x_A})^{r_A}$, where g^{x_A} is the public key agreement key of KRA_A .

A: user, B: VASP, TTP_A : TTP of A

1. $A \rightarrow B : g^{r_A}, s_A, \{A\}_L, TTP_A$
 2. $A \leftarrow B : r_B, h_2(K_{AB}, r_B, B), B_{Cert}$
 3. $A \rightarrow B : \{h_3(g^{r_A}, g^b, r_B, B)\}K_A^{-1}, A_{Cert}\}_{K_{AB}}$
- $$A, B : K_{AB} = h_1(r_B, g^{br_A})$$
- $$A : r_A = f(w_A, s_A); L = (g^{x_A})^{r_A}$$
-

Fig. 3. KR Enhanced B-Variant Protocol

A request for key recovery to KRA_A will include the public values s_A , $\{A\}_L$, r_B and g^b . With this information, KRA_A can decrypt $\{A\}_L$ and, from A's identity, obtain the corresponding secret key w_A , which in turn can be used to recompute K_{AB} . In B's domain the situation is different. B escrows his private key b with his associated agent KRA_B during the registration phase. Thus KRA_B can restore K_{AB} when presented with r_B and g^{r_A} . The requests for KR will have to be accompanied by the appropriate authorisation (warrant). Clearly, other fields from the above protocol will also have to be submitted together with the request so that the KRA can check that the request for KR is within the scope of the warrant. However, for the sake of brevity we omit such details.

As a further enhancement Rantos and Mitchell [17] point out that w_A can be a temporary secret computed using a second one-way function f^* as

$$w_A = f^*(w_A^*, TT)$$

where w_A^* is a long-term secret shared with KRA_A , and TT is a date stamp. With this enhancement, KRA_A can delegate her capability to carry out key recovery to authorised parties, such as law enforcement agents, for the periods of validity of the values w_A . Thus, when an authorised request for the interception of A's messages during a certain period is presented to KRA_A , instead of having to participate in the recovery of individual session keys, KRA_A can give the appropriate w_A values to the interception agents so that they can recover the keys themselves. This clearly makes more flexible and efficient the way in which wiretapping is performed. Unfortunately, the same does not apply to B's domain, since b is the long-term secret key of B and consequently cannot be disclosed, for it would compromise both past and future communications.

A slight variation is also described [17] that includes s_A in the encryption of the first message as

1. $A \rightarrow B : g^{r_A}, \{A, s_A\}_L, TTP_A$

We notice though that in this case the enhancement described above where w_A is a fixed term secret cannot be used. The KRA would still need to cooperate in the recovery of all session keys, for only he can obtain s_A .

The second solution proposed by the authors does not require any shared secret. Again, only the first message is changed:

$$1. A \rightarrow B : g^{r_A}, \{A, r_A\}_L, TTP_A$$

Here r_A is again a randomly chosen value. The KRA can obtain r_A directly by decrypting the second field in the message, and therefore recompute K_{AB} .

4 Analysis of the KR Enhanced ASPeCT Protocol

In this section we study the properties of KR enhanced ASPeCT protocol. In particular we investigate whether the requirements defined in Section 2 are satisfied by such protocol.

R1. Session Key Recoverability The first of the requirements is clearly achieved. If the protocol is executed correctly, both KRAs can independently restore the session key established by A and B , by proceeding as described above.

R2. Computational Overhead Since modular exponentiations are the operations that consume the most computational resources, we use the number of exponentiations as an indicative measure of the computational complexity of the protocols. Thus, we observe that for the KR enhanced B-variant protocol the introduced computational overhead consists of an extra exponentiation on A 's side in computing $L = (g^{x_A})^{r_A}$, which in turn is used to encrypt A 's identity. Since the C-variant protocol already calculated L , no extra-exponentiation is introduced in the KR enhanced version of the protocol.

The authors reason that explicitly adding A 's identity in the KR enhanced B-variant protocol allows KRA_A to obtain the corresponding secret key w_A . The identity is further encrypted under L in order to preserve the anonymity of the user. Notice that in the original protocol the user remains unidentified to B until he receives the third message. However, it is our contention that such a field is altogether unnecessary. To see this we have to realise that anonymity revocation has to occur prior to KR, when the encrypted communications are intercepted. In other words, the interception agents that want to wire-tap some specified target user's communications need to be able to discriminate such communications from the rest of communications that occur simultaneously. Once this is done, the intercepted KRI can be used to request KR from the KRA. Notice that since A 's identity is encrypted under L , which is only known to A and KRA_A , the interception agents will not be able to use that field to discriminate communications based on a target identity and, therefore, a different mechanism outside the scope of the protocol would be required.

It seems that a likely way in which the filtering would be performed is by requiring the cooperation of the NOs [11]. Recall that during the set-up phase of a communications association between A and B , both users authenticate themselves to their respective NOs. Hence NOs are the obvious candidates for filtering encrypted communications based on target identities. They could hand over the intercepted encrypted data to the authorised LEAs or, even more, interact as a proxy between the LEAs and the KRAs. In any case since the identification of the encrypted data has been performed before the actual submission of the KRI

to the KRA, the inclusion of $\{A\}_L$ in message 1 seems unnecessary, for the identity of the target user is passed by the LEA to the KRA as part of the warrant. We note that this must have also been the assumption of Rantos and Mitchell [17] when they pointed out the possibility of delegating the KR capability to authorised requesters by giving them w_A , in the case where w_A is a temporary value.

Hence, from the above argument, we see that the field $\{A\}_L$ can be safely dropped from message 1. This will save an expensive exponentiation if the first KR mechanism is used in its first variant, i.e. with s_A being sent in the clear. Thus the first message of the protocol becomes:

$$1. A \rightarrow B : g^{r_A}, s_A, TTP_A$$

Obviously when the second KR mechanism is used with the B-variant protocol L has to be computed, for r_A still must be encrypted using it.

R3. Unobtrusiveness A significant defect in the above protocol is the failure to satisfy requirement R3 for unobtrusiveness. Strictly speaking, the mutual authentication service that was originally provided is sacrificed when the KR capability is added to the WAKE protocol. User A cannot be sure whether the protocol messages are being exchanged with B or KRA_B , for both know b . Even worse, in the case where w_A is a temporary secret an impersonation attack can be mounted as explained below.

Attack An attacker C with knowledge of w_A can impersonate B to A during an authorised interception period. After intercepting message 1, C calculates $r_A = f(w_A, s_A)$, chooses a random value r'_B , and computes the session key as

$$K'_{AB} = h_1(r'_B, (g^b)^{r_A})$$

He proceeds by forming message 2 as shown in Figure 4 and sending it to A , effectively impersonating B .

A: user, C: attacker impersonating B to A, TTP_A: TTP of A

1. $A \rightarrow C : g^{r_A}, s_A, \{A\}_L, TTP_A$
 2. $A \leftarrow C : r'_B, h_2(K'_{AB}, r'_B, B), B_{Cert}$
 3. $A \rightarrow C : \{h_3(g^{r_A}, g^b, r'_B, B)\}_{K_A^{-1}}, A_{Cert}\}K'_{AB}$
- $$A, C : K'_{AB} = h_1(r'_B, g^{br_A})$$
- $$A : r_A = f(w_A, s_A); L = (g^{x_A})^{r_A}$$
-

Fig. 4. Impersonation attack in the KR Enhanced B-Variant Protocol

The above attack can be easily fixed by simply holding s_A until message 3 of the protocol. Even with this modification, R3 is not satisfied. In order to

comply with requirement R3, escrow of b has to be avoided. With this condition, we notice that the secret information that is needed by B 's KRA in order to be able to recover the session key is g^{br_A} . Hence B needs to somehow make possible KRA_B 's access to such piece of data. At first thought, we may suggest to convey it in message 2 of the protocol. For example, B generates r_B in the same way as A , i.e. $r_B = f_B(w_B, s_B)$, where w_B is a secret value shared with KRA_B , and s_B is a random number; and then sends $r_B \oplus g^{br_A}$ and s_B in message 2. However the same kind of impersonation attack can be mounted yet again. When an interception agent with temporary knowledge of w_B sees messages 2, he intercepts it and extracts g^{br_A} by first calculating

$$r_B = f_B(w_B, s_B)$$

and therefore

$$g^{br_A} = r_B \oplus (r_B \oplus g^{br_A}).$$

Now he can generate a new random number r'_B , and impersonate B by generating a new bogus message and sending it to A . Hence we see that B cannot give his KRA access to g^{br_A} in message two. But message two is the only one that he produces in the protocol. An obvious solution entails sending a fourth message with the involved communications overhead. Alternatively, we can require the cooperation of A . For example, instead of sending s_B in the clear, B could send $\{s_B\}_{K_{AB}}$ in message two. On receiving it, A would decrypt s_B and include it in message 3. Figure 5 shows a modified version of the KR enhanced B-variant protocol with all the suggested alterations. Notice that s_A is withheld until message 3 to counteract the first impersonation attack described above. Also, we have dropped the encryption of A 's identity under L in message 1 in accordance with our discussion on requirement R2.

A: user, B: VASP, TTP_A : TTP of A

1. $A \rightarrow B : g^{r_A}, TTP_A$
2. $A \leftarrow B : r_B \oplus g^{br_A}, h_2(K_{AB}, r_B, B), \{s_B\}_{K_{AB}}, B_{Cert}$
3. $A \rightarrow B : \{\{h_3(g^{r_A}, g^b, r_B, B)\}_{K_A^{-1}}, A_{Cert}\}_{K_{AB}}, s_A, s_B$

$A, B : K_{AB} = h_1(r_B, g^{br_A})$

$A : r_A = f_A(w_A, s_A)$

$B : r_B = f_B(w_B, s_B)$

Fig. 5. Modified KR Enhanced B-Variant Protocol

When A receives the second message she extracts r_B , using the value $(g^b)^{r_A}$. With r_B she can compute K_{AB} and obtain s_B by decrypting the second last field of message 2. The authentication of B to A is finalised when A checks that the hash value of message 2 corresponds to the value that she has calculated for K_{AB} .

R4. Fine Granularity We can easily verify that the requirement for fine granularity is achieved by the KR enhanced protocols. In the case where w_A is a long term secret, the keys are recovered by the KRAs in an individual basis in both domains, and the recovery of any subset of them does not compromise any other session key. In the case where w_A is a temporary secret in A 's domain, the period of validity of w_A can be adjusted to yield any convenient granularity.

5 Enforceability

Enforceability refers to the safeguards that a KR scheme might provide in order to ensure that the KR mechanism cannot be circumvented by the users of the scheme. This could happen if users succeed in communicating confidentially without allowing the recovery of the relevant session key. The actual scope of any enforceability mechanism depends on the threat model used for the KR system. Thus, we can distinguish two different enforceability levels:

Level 1 At this level, the enforceability mechanism ensures that no user of the system (including VASPs) can succeed in circumventing the KR mechanism unilaterally, i.e. without the cooperation of the other communicating party.

Level 2 At this level, the enforceability mechanism ensures that no user of the system (including VASPs) can succeed in circumventing the KR mechanism, even with the cooperation of the other communicating party.

In the ASPeCT protocol, if user A does not follow the protocol accurately and sends a bogus value as the second field in the first message, then the session key will not be recoverable in A 's domain. This stems from the lack of enforceability of the KR mechanism in A 's domain. The same cannot be said of B 's domain, at least at a level 1 enforceability. Here, KR is enforced by the certification process itself. B is refused the certification of his public key agreement key if the corresponding private key is not escrowed.

When the KR mechanisms are used with the C-variant protocol, Rantos and Mitchell [17] point out that, if required, B can relay the first message to A 's on-line TTP who then validates that the correct KR field was sent by reconstructing g^{r_A} . This helps avoid single rogue user attacks. Thus the TTP who, let's recall, is both A 's CA and KRA, can additionally perform a KRI validation function. More specifically we define such a type of function as a *private KRI validation function* since in order to validate the KRI it is necessary to have knowledge of access-restricted information, in this case w_A .

Similarly we can define a *public KRI validation function* as a validation function that can be executed by anyone using only publicly available information. The provision of one such function would have the advantage of not requiring the involvement of the KRA. This makes that function easily distributable, shifting away the heavy burden placed on the KRAs if they were to oversee all the communications of their associated users. In the next section of this paper we present a refined KR mechanism that provides a public KRI validation function without requiring an increase in the computational load of A and B . Thus, with

the new KR mechanism, validation can be performed efficiently in both the B and C-variants of the ASPeCT protocol. Furthermore, there would be no need for the TTP to act both as the CA and the KRA, as far as KR validation is concerned.

Whether enforceability, at any degree, is a requirement for KR in the wireless communications arena is an open question. A pervasive problem with KR, which applies to all techniques, is that of super-encryption [5]. From a cryptographic point of view, once an authenticated communications channel is provided, establishment of non-recoverable keys is a trivial issue. For example, an authenticated channel is all that is required to securely run the Diffie-Hellman key agreement protocol [9], which could be executed at the application layer to establish a “parallel” session key that is not recoverable by the KRAs and which is used to encrypt the data (also at the application layer) before passing it to the communications layer where the protocols described above operate. Thus, even if the session key established at the communications layer is restored by a KRA, access to the plaintext data is still not possible. In other words, Level 2 enforceability does not appear achievable, at least without the utilisation of trusted functionality (e.g. tamperproof implementations) to thwart misuse of the system. On the other hand, even if misuse of the system is technically possible, the wiretapping capability left in practice to law enforcement agencies may still make the deployment of such KR system worthy.

6 A New KR Mechanism

As already mentioned, a more flexible option to the validation mechanism proposed in [17] would be to provide a public KRI validation function. This allows any third party to detect misuse of the protocol by using only public data. For example, in the wireless environment, NOs could be given the task to perform the validation function. Thus, if required, they could enforce the KR mechanism by aborting any communications in cases where the KRI cannot be validated successfully. It is important to realise that still, the only entities that could recover session keys would be the KRAs. In this section we describe a new KR mechanism that provides a public KRI validation function and that could be easily incorporated to the ASPeCT protocol in *A*'s domain.

Firstly we note that the new KR mechanism is applicable in cases where the target key to be recovered is of the form

$$K_{AB} = f(r, \text{other public information})$$

where f is a publicly known one-way function and r is a secret random number generated by the user. In these cases, key recovery is equivalent to restoring r .

In the description of the KR mechanism we use the following notation: p a large prime, q a prime with $q|p - 1$, and g an element in the multiplicative group \mathbb{Z}_p^* of order q . All operations are performed modulo p , except where noted otherwise. The KR mechanism consists of three stages as follows.

KRI Generation phase The user, say A does the following.

1. Generate a secret KR key w , $1 \leq w \leq q - 1$, which she shares with her associated KRA. Optionally w can be generated by the KRA, or agreed by, both parties. The value $\phi = g^w$ is made publicly available. This step can be a one-off process, or alternatively, repeated at any convenient frequency.
2. Select a random integer r , such that $1 \leq r \leq q - 1$, and compute $u = g^r$.
3. Compute $c = h(u) \bmod q$, where h is an appropriate hash function.
4. Compute $s = wc + r \bmod q$.

The KRI in A 's domain is $\{w, u, s, A\}$ of which $\{u, s, A\}$ are public.

Public KRI Validation phase Given the public input data u, s, A , a monitoring third party V can check the integrity of the KRI fields generated by a user A , by doing the following:

1. Obtain authentic public value ϕ .
2. Compute $c' = h(u) \bmod q$.
3. V resolves the validation process as successful if and only if $g^s = \phi^{c'} u$.

KR phase Provided that the KRI verification function is successful, the corresponding key recovery agent can recover the value r_i when presented with the input data $\{s, u, A\}$ by doing the following:

1. Obtain w corresponding to user A .
2. Compute $c = h(u) \bmod q$.
3. Compute $r = s - wc \bmod q$.

6.1 Security of the KR Mechanism

The security of the above mechanism can be reduced to that of the non-interactive proof of possession of $\log_g \phi_i$ as implemented in Schnorr's signature scheme [18] whose properties are well known. When user i generates $\{s_i, u_i\}$, she produces a proof of knowledge of w . Its security relies on the following two propositions:

1. Knowledge of w implies the capability of recovering r , and vice-versa.
2. Provided that h can be assumed to yield random values, knowledge of $\{\phi, u, s\}$ does not give any knowledge about w , no matter how many times we reuse ϕ to produce such triplets.

It is interesting to note that we may regard r as a *verifiable encryption* of the discrete log of the public value s . The verification is much more efficient than other more general verifiable encryption protocols [19]. The reason that this is possible is that in our case the verifier never actually obtains the value r (which would give away the shared secret w). This means that our mechanism is not applicable in applications such as fair exchange [3].

6.2 ASPeCT Protocol with New KR Mechanism

It is easy to see that the above mechanism can be used in the ASPeCT protocol by making $\phi_A = g^{w_A}$ public and changing the way r_A and s_A are calculated. Now, according to the above mechanism r_A is a random value and s_A is calculated as

$$s_A = w_A h(g^{r_A}) + r_A \bmod q$$

The rest of the protocol is the same. Now any monitoring third party can check the validity of the KRI formed by A using the KRI validation function described above. Hence, there is no need to require A 's TTP to be on-line in order to validate the KRI, which makes the above mechanism suitable for both the B and C-variants of the ASPeCT protocol.

Unfortunately, due to the different way in which B authenticates to A , the same KR mechanism cannot be applied in B 's domain in the modified KR enhanced ASPeCT protocol that we proposed in Section 4. This would not happen for other protocols which are more symmetric, such as the STS protocol [10].

7 Multiple KRAs

When designing KR schemes, it is common practice to distribute the trust vested in the KRA functionality among multiple KRAs, i.e. users have several associated KRAs that have to cooperate in order to perform KR. This helps increasing the users' acceptability on the KR system. We observe however that, contrary to most KR proposals, the ASPeCT KR scheme specifies only one KRA. Therefore, it is possible for a single KRA to wiretap on users communications. Notice that a KRA can revoke the identity of any user she is associated with in polynomial time by simply testing all the possible identities.

A simple example of how to allow multiple KRAs using the modified KR enhanced ASPeCT protocol (Figure 5) is as follows.

A user, say A , establishes a secret value w_i with each KRA, KRA_i ($i = 1, \dots, n$). For each run of the protocol, A calculates

$$r_A = f_A(w_1, s_A) \oplus f_A(w_2, s_A) \oplus \dots \oplus f_A(w_n, s_A)$$

The rest of the protocol remains the same. An authorised requester seeking KR does the following:

1. Contacts each KRI_i and presents the appropriate authorisation information, together with the values s_A and A corresponding to the intercepted communication. KRI_i then calculates $v_i = f_A(w_i, s_A)$, which she returns to the requester.
2. Once all the KRAs have been contacted, she restores r_A as:

$$r_A = f_A v_1 \oplus v_2 \oplus \dots \oplus v_n$$

3. Finally, since r_B, g^b are public information, she can compute the session key as:

$$K_{AB} = h_1(r_B, g^{b^{r_A}}).$$

8 Conclusion

In this paper, we analysed a key recovery proposal for the ASPeCT protocol and identified several weaknesses. A modification was proposed that fixes the weaknesses and, exhibits additional and improved properties.

References

1. ACTS AC095, ASPeCT Deliverable D02, Initial Report on Security Requirements, AC095/ATEA/W21/DS/P/02/B, Feb., 1997, Available from <http://www.esat.kuleuven.ac.be/cosic/aspect/>. [227](#)
2. Advanced Security for Personal Communications Technologies. <http://www.esat.kuleuven.ac.be/cosic/aspect/index.html> [226](#)
3. N.Asokan, V. Shoup and M. Waidner, Optimistic Fair Exchange of Digital Signatures, Eurocrypt'98, Springer-Verlag, 1998, pp.591-606. [235](#)
4. U. Black, Third Generation Mobile Systems (TGMSs), in Second Generation Mobile & Wireless Networks, Parentice Hall, 1999. [223](#)
5. B. Pfitzmann and M. Waidner, How to Break Fraud-Detectable Key Recovery, Operating Systems Review, 21, 1998, pp.23-28. [234](#)
6. K. Buhanal et al., IMT-2000: Service Providers Perspective, IEEE Personal Communications, August 1997. [223](#)
7. CESG, Securing Electronic Mail within HMG - part 1: Infrastructure and Protocol, document T/3113TL/2776/11, March 1996, available at <http://www.rdg.opengroup.org/public/tech/security/pki/casm/casm.htm>.
8. Denning D. and Branstad D., A Taxonomy for Key Escrow Encryption systems, Communications of the ACM, Vol. 39, Pp 34-40, 1996.
9. Diffie W. and Hellman M., New Directions in Cryptography, IEEE Transactions on Information Theory, 22, pp 644-654, 1976. [225](#) [234](#)
10. W.Diffie, P. van Oorschot and M.Wiener, Authentication and Authenticated Key Exchanges, Designs Codes and Cryptography, 2, pp.107-125, 1992. [228](#), [236](#)
11. ETSI TC-STAG, "Security Techniques Advisory Group (STAG); Definition of User Requirements for Lawful Interception of telecommunications; Requirements of the Law Enforcement Agencies", ETR 331, December 1996. [226](#), [230](#)
12. ETSI SMG10, Draft UMTS 33.21 version 2.0.0, Universal Mobile Telecommunications System (UMTS): Security Requirements, Feb., 1999. [225](#)
13. ETSI TC Security, Specification for Trusted Third Party Services: Part1 Key Management and Key Escrow/Recovery, DEN/SEC-003001x, Draft Version 1,0 (edition2), 11th, Nov. 1997 [226](#)
14. Horn G. and Preneel B., Authentication and payment in future mobile systems, Computer Security - ESORICS'98, Lecture Notes in Computer Science, 1485, pp. 277-293, 1998. [226](#), [227](#), [228](#)
15. K. Martin, Applying Cryptography within the ASPeCT Project, Information Security Technical Report, 1998, Vol. 2, No. 4, pp. 41-53. [223](#)
16. T. Ojanpera and R. Prasad, IMT-2000 Applications, in Widenband CDMA for Third Generation Mobile Communication, T. Ojanpera and R. Prasad (ed.), Artech House Publishers, 1998, pp. 65-76. [223](#)
17. Rantos K. and Mitchell C., Key recovery in ASPeCT authentication and initialisation of payment protocol, Proceedings of ACTS Mobile Summit, Sorrento, Italy, June 1999. [226](#), [228](#), [229](#), [231](#), [233](#), [234](#)
18. Schnorr C.P., Efficient Identification and Signatures for Smart Cards- CRYPTO'89, Proceedings, Lecture Notes in Computer Science, vol 330, Springer-Verlag, pp 239-251, 1990. [235](#)
19. M. Stadler, Publicly Verifiable Secret Sharing, Eurocrypt'96, Springer-Verlag, 1996, pp.190-199. [235](#)
20. US Department of Commerce, National Institute of Standard and Technology, FIPS PUB 185, Escrowed Encryption Standard, February 1994.

Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications

Katsuyuki Okeya^{1,*}, Hiroyuki Kurumatani^{1,*}, and Kouichi Sakurai²

¹ Hitachi, Ltd., Software Division,

Kaneichi Bldg. 549-6, Shinano-cho, Totsuka-ku, Yokohama, 244-0801, Japan

{okeya_k, kurumahi}@soft.hitachi.co.jp

² Kyushu University,

Department of Computer Science and Communication Engineering

6-10-1, Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan

sakurai@csce.kyushu-u.ac.jp

Abstract. We show that the elliptic curve cryptosystems based on the Montgomery-form $E^M : BY^2 = X^3 + AX^2 + X$ are immune to the timing-attacks by using our technique of randomized projective coordinates, while Montgomery originally introduced this type of curves for speeding up the Pollard and Elliptic Curve Methods of integer factorization [Math. Comp. Vol.48, No.177, (1987) pp.243-264].

However, it should be noted that not all the elliptic curves have the Montgomery-form, because the order of any elliptic curve with the Montgomery-form is divisible by “4”. Whereas recent ECC-standards [NIST,SEC-1] recommend that the cofactor of elliptic curve should be no greater than 4 for cryptographic applications.

Therefore, we present an efficient algorithm for generating Montgomery-form elliptic curve whose cofactor is exactly “4”. Finally, we give the exact condition on the elliptic curves whether they can be represented as a Montgomery-form or not. We consider divisibility by “8” for Montgomery-form elliptic curves.

We implement the proposed algorithm and give some numerical examples obtained by this.

Keywords: Elliptic Curve Cryptography, Montgomery-form, Efficient Implementation, Timing-attacks

1 Introduction

We consider the exact condition on the elliptic curves whether they can be represented a Montgomery-form or not, and present an efficient algorithm for generating Montgomery-form elliptic curves whose cofactor is exactly “4”. We also implement the algorithm and give some numerical examples obtained by this.

* Okeya and Kurumatani are supported by Information-technology Promotion Agency, Japan (IPA).

1.1 Elliptic Curves with the Montgomery-Form

Montgomery introduced the non-standard form $E^M : BY^2 = X^3 + AX^2 + X$ of elliptic curves in [Mon87], while the most standard form of elliptic curves is $E : y^2 = x^3 + ax + b$, which is called the Weierstrass-form.

1.2 A New Application: Preventing Timing-Attacks

We observe that the elliptic curve cryptosystems based on the Montgomery-form $E^M : BY^2 = X^3 + AX^2 + X$ are immune against timing-attacks [Koc, Koc96].

Kocher [Koc, Koc96] presented the timing-attacks: Attackers carefully measure the amount of time required to perform the private key operations, so that they might be able to decide fixed Diffie-Hellman exponents. This attack could be applicable to the elliptic curve cryptosystems including ECDSA ([ANSI]).

Time required to perform the conventional scalar multiplication algorithm based on the Weierstrass-form depends on the bit-patterns (and on the ratio between the number of zeros and the number of ones) of the secret value.

Whereas we show that the scalar multiplication on the Montgomery-form elliptic curve does *not* depend on the bit-patterns (nor on the ratio between the number of zeros and the number of ones) of the secret value. It has exactly seven multiplications and four square-multiplications on \mathbf{F}_p per bit. This is due to the specific algorithm for computing scalar multiplication nP from P , which repeatedly calculates either $(2mP, (2m+1)P)$ or $((2m+1)P, (2m+2)P)$ from $(mP, (m+1)P)$ in the Montgomery-form elliptic curves.

The computation via by choosing a representative in the projective coordinates randomly is also useful for making it more difficult to measure the amount of time required. We compute the scalar d multiplications on the affine coordinates (x, y) via a corresponding projective coordinates (kx, ky, k) , where k is randomly choosed.

Thus, Montgomery-form elliptic curves are shown to be useful for public-key cryptosystems from the point of view of not only efficient implementation but also protection against timing-attacks.

1.3 Montgomery-Form has Cofactor 4

However, the class of Montgomery-form is restricted in the elliptic curves. We should note that the order of elliptic curve with the Montgomery-form is always divisible by “4” as remarked in [Mon87]. Therefore, not all elliptic curves have a Montgomery-form.

Whereas recent ECC-standards [NIST99, SEC-1] recommend that the cofactor of elliptic curves be within “4” for cryptographic use. Thus, we shall design Montgomery-form elliptic curves with cofactor exactly “4” for ECC-standards [NIST99, SEC-1].

1.4 Our Criteria and Generating Algorithm

We consider transformability of elliptic curves from a Weierstrass-form to a Montgomery-form, and give exact condition on the elliptic curves whether they can be represented as a Montgomery-form or not. For checking whether its cofactor is exactly “4”, we further consider divisibility by powers of 2 for the curve orders of the Montgomery-form elliptic curves. In particular, the discussion of divisibility by 8 is the most significant.

Using our criteria, we present an efficient algorithm for generating Weierstrass-form elliptic curves with Montgomery-form and with which cofactor is equal exactly to 4. Our algorithm handles not only the original curve itself but also its twist so that it can find the good curve more efficiently. We also implement our algorithm by using Schoof’s order-counting algorithm, then experimentally confirm the validity of our algorithm. In fact, our algorithm has produced many curves with cryptographic properties desireble for practical applications.

We should note that in this paper we mainly discuss on the elliptic curves over prime fields. However, the similar argument can be applicable to any elliptic curves over any finite fields including Optimal Extension Fields (OEF) [BP98, MKKH99].

2 Preliminaries

In this section, we define technical terms for the following sections. Let $p(\geq 5)$ be a prime and \mathbf{F}_p be the finite field of order p . For $A, B \in \mathbf{F}_p$, an elliptic curve defined by

$$E^M : BY^2 = X^3 + AX^2 + X$$

is called a Montgomery-form elliptic curve or a Montgomery-type elliptic curve. For numbers $a, b \in \mathbf{F}_p$, an elliptic curve defined by

$$E : y^2 = x^3 + ax + b$$

is called a Weierstrass-form elliptic curve or a Weierstrass-type elliptic curve. The set of (\mathbf{F}_p -rational) points of E or E^M forms a group with the point at infinity \mathcal{O} as the identity element. Refer to the next section for additional-operation formulae on the Montgomery-form elliptic curves. The number of points of E (resp. E^M) is called curve orders and denoted by $\#E$ (resp. $\#E^M$). For a point P on an elliptic curve, (point) order is the least positive integer n such that $nP = \mathcal{O}$. For example, the point $(0, 0)$ on any Montgomery-form elliptic curve is of order 2. Cofactor is the quotient of the curve order divided by the base point order.

Let $r \in \mathbf{F}_p$ be quadratic non-residue. For a Weierstrass-form elliptic curve $E : y^2 = x^3 + ax + b$,

$$E_r : y^2 = x^3 + ar^2x + br^3$$

is called a twist of E and for a Montgomery-form elliptic curve $E^M : BY^2 = X^3 + AX^2 + X$,

$$E_r^M : \frac{B}{r}Y^2 = X^3 + AX^2 + X$$

is called a twist of E^M . It is clear that $\#E + \#E_r = 2(p+1)$ and $\#E^M + \#E_r^M = 2(p+1)$.

We define a Weierstrass-form elliptic curve E as transformable to the Montgomery-form, if there exists a Montgomery-form elliptic curve defined over \mathbf{F}_p $E^M : BY^2 = X^3 + AX^2 + X$ such that E and E^M are isomorphic over \mathbf{F}_p . Namely, there exists $s, t, \alpha, \beta \in \mathbf{F}_p, s, t \neq 0$ such that the function mapping $(x, y) \in E(\mathbf{F}_p)$ to $(s(x - \alpha), t(y - \beta))$ is a group isomorphism of $E(\mathbf{F}_p)$ and $E^M(\mathbf{F}_p)$. $\#E = \#E^M$ if E is transformable to E^M .

3 Cryptographic Advantages of Montgomery-Form Elliptic Curves

3.1 A Comparison between the Montgomery-Form and the Weierstrass-Form about the Operations

The operations on the Montgomery-form elliptic curve $E^M : BY^2 = X^3 + AX^2 + X$ for affine coordinates are as follows. Let $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ be points on E^M . Then, the point $P_3 = (x_3, y_3) = P_1 + P_2$ is the following:

addition formulae ($P_1 \neq \pm P_2$)

$$\begin{aligned} \Lambda &= (y_2 - y_1)/(x_2 - x_1) \\ x_3 &= B\Lambda^2 - A - x_1 - x_2 \\ y_3 &= \Lambda(x_1 - x_3) - y_1 \end{aligned}$$

doubling formulae ($P_1 = P_2$)

$$\begin{aligned} \Lambda &= (3x_1^2 + 2Ax_1 + 1)/(2By_1) \\ x_3 &= B\Lambda^2 - A - 2x_1 \\ y_3 &= \Lambda(x_1 - x_3) - y_1 \end{aligned}$$

Next, we set $(x, y) = (X/Z, Y/Z)$ for a point (x, y) on E^M , and give operations on projective coordinates. The n -times point of a point $P = (X, Y, Z)$ is denoted by $nP = (X_n, Y_n, Z_n)$. According to [Mon87], $(m+n)P = mP + nP$ without Y is as follows.

addition formulae ($m \neq n$)

$$\begin{aligned} X_{m+n} &= Z_{m-n}[(X_m - Z_m)(X_n + Z_n) + (X_m + Z_m)(X_n - Z_n)]^2 \\ Z_{m+n} &= X_{m-n}[(X_m - Z_m)(X_n + Z_n) - (X_m + Z_m)(X_n - Z_n)]^2 \end{aligned}$$

doubling formulae ($m = n$)

$$\begin{aligned} 4X_nZ_n &= (X_n + Z_n)^2 - (X_n - Z_n)^2 \\ X_{2n} &= (X_n + Z_n)^2(X_n - Z_n)^2 \\ Z_{2n} &= (4X_nZ_n)((X_n - Z_n)^2 + ((A+2)/4)(4X_nZ_n)) \end{aligned}$$

The addition formulae require four multiplications and two squarings on \mathbf{F}_p and the doubling formulae require three multiplications and two squarings on \mathbf{F}_p .

The scalar multiplication nP requires us repeatedly to calculate either $(2mP, (2m+1)P)$ or $((2m+1)P, (2m+2)P)$ from $(mP, (m+1)P)$ depending on each bit of binary digit of n . Put k the bit length of n . Then the repeating time is $(k-2)$. Without loss of generality, we can assume $Z_1 = 1$. So, addition

formulae require three multiplications. It needs to compute $2P$ at first, the computation time of nP is $(3M + 2S)(2k - 3)$, where M is the computation time of the multiplications and S is the computation time of the squarings on the finite field.

On the scalar multiplications on Weierstrass-form elliptic curves, Jacobian coordinates using window method is the fastest ([CMO98]). Assume that the size of definition field is 160 bits and that $1S = 0.8M$. The scalar multiplications on Weierstrass-form require $10M$ per bit on average. The scalar multiplications on Montgomery-form require $9.2M$ per bit. Thus, the Montgomery-form elliptic curves are faster than the Weierstrass-form elliptic curves by about 10 percent.

Remark 1. There are more detailed comparisons between the computation times of the Montgomery-form and those of the Weierstrass-form in [TK99, Izu99a, Izu99b, OSK99].

3.2 The Montgomery-Form Elliptic Curves against Timing-Attacks

A timing-attack is a way of guessing a private key information from its calculating time of operation on cryptosystems like the RSA and DSS([Koc, Koc96]), and is adaptable to elliptic curve cryptosystems. In the case of elliptic curve cryptosystem, it is a way of guessing a private key d from the calculating time of the scalar multiplication dP of a base point P by d . It is effective that the calculating time is far from the average time.

The number of additions and that of doublings in the scalar multiplications on the Montgomery-form elliptic curves depend just on the bit-lengths but do *not* depend on the bit-patterns (nor on the ratio between the number of zeros and the number of ones): in the previous section, we see that the specific algorithm for computing the scalar multiplications dP on the Montgomery-form elliptic curves using projective coordinates repeatedly calculate either $(2mP, (2m+1)P)$ or $((2m+1)P, (2m+2)P)$ from $(mP, (m+1)P)$ depending on a certain bit. Of course, the point $2mP$ is mP doubled, the point $(2m+1)P$ is mP added by $(m+1)P$, and the point $(2m+2)P$ is $(m+1)P$ doubled. Thus, the scalar multiplication requires *one addition on the elliptic curve* and *one doubling on the elliptic curve* per bit. The number of additions and that of doublings, which are just one respectively, do *not* depend on whether the certain bit is 0 or 1.

However, on the Weierstrass-form elliptic curve, the number of additions and that of doublings in the scalar multiplications *depend* on the bit-patterns (and on the ratio between the number of zeros and the number of ones): for computing the scalar multiplications dP , it needs to calculate repeatedly either $2mP$ or $(2m+1)P$ from mP (when using window method, it is more complicated but the following result is almost same). Since $2mP = 2(mP)$ and $(2m+1)P = P + 2(mP)$, the scalar multiplication requires one doubling on the elliptic curve, or one doubling and one addition on the elliptic curve depending on whether the certain bit is 0 or 1. The computation time in the case that the certain bit is 0 is shorter than that in the case that it is 1 by one addition on the elliptic curve. Assume that the scalar value d has many zeros as compared with

ones. In that case the number of additions and that of doublings in the scalar multiplication are small, and its computation time is far from the average time. Hence, timing-attacks are effective for such values.

Elliptic curves defined over finite fields with characteristic 2 using scalar multiplications like the Montgomery-form also immune to timing-attacks. Refer to [AMV93] for the scalar multiplications with characteristic 2 like the Montgomery-form. We need to pay attention that the assumptions in [AMV93] for deriving scalar multiplications are not in general. That is, we may not assume that the z -coordinate of any $(2m + 1)P$ is equal to 1. (See Appendix A for the detailed descriptions.)

3.3 Further Improvement: Randomized Projective Coordinates

In the previous section, we saw that the Montgomery-form elliptic curves have the advantage of immunity to timing-attacks. In this section, we propose further improvement for preventing timing-attacks — *randomized projective coordinates*.

The number of additions and that of doublings in the scalar multiplications on the Montgomery-form elliptic curves are constant, but the computation times of the additions and those of the doublings on the elliptic curves are *not* constant. This is because the additions on the elliptic curves require four multiplications and two squarings on the finite field which is the definition field of the elliptic curves, and the doublings on the elliptic curves require three multiplications and two squarings on the finite field, and the multiplication/squaring on the finite field has discrepancies among its computation times although the number required of multiplications and that of squarings on the finite field for the addition/doubling on the elliptic curve are constant. For values which extremely small compared with the characteristic p of finite fields, the computation times of multiplication/squaring are short. Consequently, computation times of addition/doubling are short for points having such values. And the computation time of the scalar multiplication is short comparatively if there are such points in the calculating of the scalar multiplication. Therefore, the computation time of the scalar multiplication on the elliptic curve depends on *the operations on the finite field*. The same values have the same time required for computation, and it is easy for us to guess which values have the time required for computation far short/long from the average. The fact mentioned above gives an information for timing-attacks.

We present *randomized projective coordinates* for avoiding the situation above:

INPUT A scalar value d and a base point $P = (x, y)$.

OUTPUT The scalar multiplication dP .

1. Generate a random number k .
2. Calculate $P = (kx, ky, k)$ expressed by projective coordinates.
3. Calculate dP using the scalar multiplication algorithm with projective coordinates on the Montgomery-form elliptic curve.
4. Output dP .

Since $(kx, ky, k) = (x, y, 1)$ in projective coordinates, the computation result is coincide with the result using $(x, y, 1)$, which is usual choice. The computation times using random numbers for the same value are different. Some of them may be short, but they are not always short. This fact prevents timing-attacks.

Remark 2. Using only randomized projective coordinates is not good enough for preventing timing-attacks. (See [Cor99] for preventing Differential Power Analysis (DPA) by using randomized projective coordinates.) On the computation of the scalar multiplication dP of a point P by a private key d on the Weierstrass-form, the number of additions and that of doublings are *proper* to the private key d (and the number of additions and that of doublings in the scalar multiplication using window-methods are also proper to d). That is, the number of additions and that of doublings for another private key d' are different from those of d , in general. Therefore, an adversary repeatedly obtains computation times for the same point, and he can estimate the number of additions and that of doublings on the elliptic curve for the private key by statistical treatment for the distribution of the computation times.

On the other hand, in the case of the Montgomery-form, the number of additions and that of doublings on the elliptic curve are constant for the same bit length private keys. Thus, if we could assume that the computation time of additions and that of doublings on the elliptic curve are constant for any point, the computation time of the scalar multiplications are constant. However, a close situation appears by using randomized projective coordinates.

4 Transformability from Weierstrass-Form to Montgomery-Form

In this section, we study transformabilities from the Weierstrass-form to the Montgomery-form. Any Montgomery-form elliptic curve has the point $(0, 0)$ of order 2. It is easy to find that there exists a Weierstrass-form elliptic curve without the Montgomery-form, since some Weierstrass-form elliptic curves have no points of order 2. The Weierstrass-form elliptic curves with the Montgomery-form should have the point of order 2 which is mapped to $(0, 0)$ on the Montgomery-form elliptic curves. In fact, they are transformable to the Montgomery-form if they have such a point. The next proposition ensures that.

Proposition 1. *A Weierstrass-form elliptic curve $E : y^2 = x^3 + ax + b$ is transformable to the Montgomery-form if and only if it satisfies two conditions as follows:*

1. *The equation $x^3 + ax + b = 0$ has at least one root in \mathbf{F}_p*
2. *The number $3\alpha^2 + a$ is quadratic residue in \mathbf{F}_p , where α is a root of the equation $x^3 + ax + b = 0$ in \mathbf{F}_p .*

Proof. Assume that E satisfies such conditions. Let s be one of the square roots of $(3\alpha^2 + a)^{-1}$ in \mathbf{F}_p , and set $B = s, A = 3\alpha s$. Then, the function mapping point (x, y) on E to $(s(x - \alpha), sy)$ gives an isomorphism E to E^M , where E^M is the Montgomery-form elliptic curve defined by $BY^2 = X^3 + AX^2 + X$.

Conversely, assume that the Weierstrass-form elliptic curve E is transformable to a Montgomery-form elliptic curve $E^M : BY^2 = X^3 + AX^2 + X$. Then, the Weierstrass-form elliptic curve should have points of order 2 in \mathbf{F}_p . Thus, the condition 1 is satisfied.

The isomorphism from the Weierstrass-form elliptic curve to the Montgomery-form elliptic curve is given that (x, y) maps to $(s(x - \alpha'), t(y - \beta'))$ for some $s, t, \alpha', \beta' \in \mathbf{F}_p, s, t \neq 0$. Since the point $(\alpha, 0)$ of order 2 on the Weierstrass-form elliptic curve corresponds to the point $(0, 0)$ on the Montgomery-form elliptic curve, we get $\alpha' = \alpha, \beta' = 0$. So, the isomorphism maps to $(s(x - \alpha), ty)$. This point is on the Montgomery-form elliptic curve. We obtain

$$Bt^2y^2 = s^3(x - \alpha)^3 + As^2(x - \alpha)^2 + s(x - \alpha). \quad (1)$$

For simplicity, set $f(x) = x^3 + ax + b$. Since the point (x, y) is on the Weierstrass-form elliptic curve, substitute $y^2 = f(x)$ at the formula (1), we find $Bt^2 = s^3$ by comparing x^3 -terms. We obtain

$$s^2f(x) = s^2(x - \alpha)^3 + As(x - \alpha)^2 + (x - \alpha), \quad (2)$$

by substituting $Bt^2 = s^3$ and dividing by s at the formula (1).

$$s^2f'(\alpha) = 1 \quad (3)$$

is derived from the formula (2) with derivation by x and substitution of α for x . Thus, $f'(\alpha)$ should be quadratic residue in \mathbf{F}_p , and the condition 2 is satisfied. \square

Remark 3. Any Montgomery-form elliptic curve is transformable to the Weierstrass-form elliptic curve. For the Montgomery-form elliptic curve $E^M : BY^2 = X^3 + AX^2 + X$, we set $s = B, \alpha = A/3B, a = 1/s^2 - 3\alpha^2, b = -\alpha^3 - a\alpha$. Then, the Weierstrass-form elliptic curve $E : y^2 = x^3 + ax + b$ is transformable to E^M .

Remark 4. There are other claims which decide whether the Weierstrass-form elliptic curves are transformable to the Montgomery-form elliptic curves or not ([Izu99a, Izu99b]). The above proposition is easy to handle in random elliptic curves generation.

Example 1. $p = 5, y^2 = x^3 + 2x$.

Since the equation $x^3 + 2x = 0$ has one root $\alpha = 0$ in \mathbf{F}_5 , Condition 1 of Proposition 1 is satisfied. However, the number $3\alpha^2 + a (= 2)$ is quadratic non-residue. Thus, this curve is not transformable to the Montgomery-form.

Example 2. $p = 7, y^2 = x^3 + 3x + 6$.

Since the equation $x^3 + 3x + 6 = 0$ has one root $\alpha = 3$ in \mathbf{F}_7 , Condition 1 is satisfied, and the number $3\alpha^2 + a (= 2)$ is quadratic residue. Thus, this curve is transformable to the Montgomery-form. $s = 2$ is one of square roots of $1/2$ in \mathbf{F}_7 . We obtain the numbers $B = s = 2, A = 3\alpha s = 4$. Hence, the Montgomery-form elliptic curve is the equation $2Y^2 = X^3 + 4X^2 + X$, and the point (x, y) on the Weierstrass-form elliptic curve $y^2 = x^3 + 3x + 6$ corresponds to the point $(2(x - 3), 2y)$ on the Montgomery-form elliptic curve $2Y^2 = X^3 + 4X^2 + X$.

Proposition 2. *Let r be quadratic non-residue in \mathbf{F}_p , and $E_r : y^2 = x^3 + ar^2x + br^3$ be the twist of $E : y^2 = x^3 + ax + b$. Then, E is transformable to the Montgomery-form if and only if E_r is transformable to the Montgomery-form.*

Proof. Assume that E is transformable to the Montgomery-form. According to Proposition 1, There exists $\alpha \in \mathbf{F}_p$ such that $f(\alpha) = 0$ and that $f'(\alpha)$ is quadratic residue, where $f(x) = x^3 + ax + b$. Set $f_r(x) = x^3 + ar^2x + br^3$. $f_r(r\alpha) = r^3f(\alpha) = 0$, and $f'_r(r\alpha) = r^2f'(\alpha)$, so it is quadratic residue. According to Proposition 1, E_r is also transformable to a Montgomery-form.

Conversely, assume that E_r is transformable to the Montgomery-form. Since r^{-1} is quadratic non-residue in \mathbf{F}_p , the elliptic curve E is the twist of E_r . As above, E is transformable to the Montgomery-form. \square

Example 3. The integer 3 is quadratic non-residue in \mathbf{F}_7 . The elliptic curve $y^2 = x^3 + 6x + 1$ is the twist of the elliptic curve $y^2 = x^3 + 3x + 6$. the number $\alpha = 2$ is the root of the equation $x^3 + 6x + 1 = 0$ and the number $3\alpha^2 + 6 = 4$ is quadratic residue. Thus, the curve is transformable to the Montgomery-form. On the other hand, we know that the elliptic curve $y^2 = x^3 + 3x + 6$ is transformable to the Montgomery-form by Example 2. Proposition 2 shows us the twist $y^2 = x^3 + 6x + 1$ is also transformable to the Montgomery-form.

According to Proposition 2, the transformabilities of a given Weierstrass-form elliptic curve and of its twist coincide with each other. When we generate elliptic curves randomly, we need to decide curve orders for judging the securities of the curves. Ordinarily, we do that for an elliptic curve candidate and its twist at the same time, since the relation $\#E + \#E_r = 2(p + 1)$ gives us that one curve order drives from the other curve order. When we generate the Weierstrass-form elliptic curves with the Montgomery-form, we can deal with a candidate and its twist together because of the coincidence of their transformabilities.

Let Δ be the discriminant of the polynomial $f(x) = x^3 + ax + b$, namely, $\Delta = -16(4a^3 + 27b^2)$. The definition of discriminant gives the following:

- The equation $f(x) = 0$ has three roots in $\mathbf{F}_p \Rightarrow (\Delta/p) = 1$
- The equation $f(x) = 0$ has one root in $\mathbf{F}_p \Rightarrow (\Delta/p) = -1$

Here (\cdot/\cdot) denotes the quadratic residue symbol. Let α, β and γ be roots of the equation $f(x) = 0$ in the algebraic closure of \mathbf{F}_p or a suitable extension field of \mathbf{F}_p . It is easy to find the equation $\Delta = -16(3\alpha^2 + a)(3\beta^2 + a)(3\gamma^2 + a)$ by calculation. Using relations above, we easily find many conditions for

transformability such as the following: When $p \equiv 1 \pmod{4}$, if the equation $f(x) = 0$ has three roots, the Weierstrass-form elliptic curve defined by the equation $y^2 = f(x)$ is transformable to the Montgomery-form.

5 Divisibilities by Powers of 2 for Curve Orders of the Montgomery-Form

Montgomery mentioned the divisibilities by “4” for curve orders of Montgomery-form in his paper ([Mon87]). According to this paper, the curve orders with the Montgomery-form are always divisible by 4. Whereas recent ECC-standards ([NIST99, SEC-1]) recommend that the cofactor of elliptic curve be within “4” for cryptographic use. For generating Montgomery-form elliptic curves with cofactor 4, we need to study the divisibilities by integers for curve orders, especially by “8”.

The next proposition and corollary describe the divisibilities by 4 for curve orders.

Proposition 3. Let $E^M : BY^2 = X^3 + AX^2 + X$ be the Montgomery-form elliptic curve. Then, E^M has :

1. three points of order 2 if $A^2 - 4$ is quadratic residue
 2. exactly one point of order 2, which is $(0, 0)$ if $A^2 - 4$ is quadratic non-residue
 3. the points $(1, \pm\gamma)$ of order 4 if $(A+2)/B$ is quadratic residue
 4. the points $(-1, \pm\gamma')$ of order 4 if $(A-2)/B$ is quadratic residue,
where γ is one of the quadratic roots of $(A+2)/B$ and γ' is one of the quadratic
roots of $(A-2)/B$.

Proof. The elliptic curve E^M always has the point $(0, 0)$ of order 2. The equation $X^2 + AX + 1 = 0$ has two roots in \mathbf{F}_p if the discriminant of the equation $A^2 - 4$ is quadratic residue. Hence, E^M has two other points of order 2 (1.). The equation $X^2 + AX + 1 = 0$ has no roots in \mathbf{F}_p if the discriminant $A^2 - 4$ is quadratic non-residue (2.). If $(A+2)/B$ is quadratic residue, the double points of the points $(1, \pm\gamma)$ are both $(0, 0)$. Thus, they are points of order 4 (3.). The case that $(A-2)/B$ is quadratic residue is similar (4.). \square

Corollary 1. The curve orders of the Montgomery-form are always divisible by 4 ([Mon87]). Thus, any Montgomery-form elliptic curve has the cofactor which is greater than or equal to 4.

Proof. First, we assume that the discriminant $A^2 - 4$ is quadratic residue. Then, the curve has three points of order 2 and the curve order is divisible by 4. Next, we assume that the discriminant $A^2 - 4$ is quadratic non-residue. Then either $(A + 2)/B$ or $(A - 2)/B$ is quadratic residue. Thus, the curve has a point of order 4 and the curve order is divisible by 4. \square

Remark 5. The book “Elliptic Curves in Cryptography”, which was recently published ([BSS99]), has many numerical examples of elliptic curves. The following elliptic curve is in the Example 11 at p.185, a Weierstrass-form elliptic

curve with cofactor 4.

$p = 000045e1\ 8f0df0d6\ ed244807\ b126feeb\ c1eab4de\ c8263bdd\ 6dc120d1\ e36b6cb5\ d7114f5d\ 883276d0\ e29dad93\ bcb542dd\ ed75343f$

$a = 00000005$

$b = 00002655\ 4794e358\ 360936a7\ 3a77d75b\ e7d64d49\ 13a8f5d1\ 7354a69b\ 3423929a\ 57f98a1d\ b34c1563\ beb79dff\ 0d40b990\ 5062b347$

The equation $x^3 + ax + b = 0$ has three roots in \mathbf{F}_p . Thus, Condition 1 of Proposition 1 is satisfied. Let α, β and γ be three roots. That is,

$\alpha = 0000195b\ 9279f672\ f0a52665\ f24df394\ 812aa7e3\ da3e8816\ 1603b4b2\ 7de839f5\ 0d1b79ad\ ac86d1c2\ 99b2501e\ 18663a2b\ af699cad$

$\beta = 00003c74\ de1cde6b\ 718e6bb6\ 622f43f9\ 5ec725f6\ b2f47967\ cd03535c\ 5b1db420\ 15d739d7\ 10bef858\ 585767b0\ 502b0d90\ e97372db$

$\gamma = 000035f2\ ad850ccf\ 7814fdf3\ 0dd0c649\ a3e39be3\ 0319763c\ f87b3994\ edd0eb56\ 8b2feb36\ 531f2386\ d331a359\ 10d93dff\ 420d58f6.$

The number $3\alpha^2 + a$

$= 0000310f\ 1870d004\ 25388cb9\ 418695a8\ ff533216\ 056c5463\ cad7fff7\ ebac7eae\ 8e620c5e\ b5027d67\ 2bae606e\ e3aa6419\ 74131b4b$

is quadratic non-residue in \mathbf{F}_p , the root α does not satisfy Condition 2. the number $3\beta^2 + a$ and the number $3\gamma^2 + a$ are also quadratic non-residue. Hence, no roots satisfy Condition 2. Therefore, this Weierstrass-form elliptic curve is not transformable to the Montgomery-form elliptic curve, although it has cofactor 4.

That is, not all the Weierstrass-form elliptic curves, of which curve orders are divisible by 4, are transformable to the Montgomery-form elliptic curves.

Concerning the divisibilities by 8 for the curve orders, we obtain the following:

Theorem 1.

$p \equiv 1 \pmod{4} \quad ((-1/p) = 1)$

$A + 2$	$A - 2$	B	$8 \#E^M$
QNR	QR	QR	D
QNR	QR	QNR	ND
QR	QNR	QR	D
QR	QNR	QNR	ND
QR	QR	QR	D
QR	QR	QNR	ND
QNR	QNR	QR	ND
QNR	QNR	QNR	D

QR :quadratic residue

QNR :quadratic non-residue

$p \equiv 3 \pmod{4} \quad ((-1/p) = -1)$

$A + 2$	$A - 2$	B	$8 \#E^M$
QNR	QR	QR	ND
QNR	QR	QNR	ND
QR	QNR	QR	D
QR	QNR	QNR	D
QR	QR	QR	D
QR	QR	QNR	D
QNR	QNR	QR	D
QNR	QNR	QNR	D

D :divisible by 8

ND :non-divisible by 8

Proof. In the case that $A^2 - 4$ is quadratic non-residue, it is a consequence of Theorem 2 below because there is just one point of order 2 on the curve. In the case that $A^2 - 4$ is quadratic residue, it is a consequence of Proposition 3 and Proposition 4 below. That is, the curve order of either the given Montgomery-form elliptic curve or its twist is divisible by 8 since either of them has a point of order 4 from Proposition 3, and we obtain the other divisibility from Proposition 4. \square

Assume that any probability of quadratic residue in Theorem 1 is exactly 1/2 and that properties of $A + 2$ and $A - 2$ for quadratic residue are independent. Then probabilities that the curve orders of the Montgomery-form are divisible by 8 are as follows.

$$\begin{array}{ll} 1/2 & \text{if } p \equiv 1 \pmod{4} \\ 3/4 & \text{if } p \equiv 3 \pmod{4} \end{array}$$

Thus, We can discard certain ratio of the Montgomery-form elliptic curves at the first stage of random elliptic curves generation.

The next theorem concerns the existence or non-existence of the points of order 8.

Theorem 2. Let $A^2 - 4$ be quadratic non-residue.

$$p \equiv 1 \pmod{4} \quad ((-1/p) = 1) \quad p \equiv 3 \pmod{4} \quad ((-1/p) = -1)$$

$A+2$	$A-2$	B	u	order 8
QNR	QR	QR	-1	E
QNR	QR	QNR	1	NE
QR	QNR	QR	1	E
QR	QNR	QNR	-1	NE

QR:quadratic residue

QNR:quadratic non-residue

$A+2$	$A-2$	B	u	order 8
QNR	QR	QR	-1	NE
QNR	QR	QNR	1	NE
QR	QNR	QR	1	E
QR	QNR	QNR	-1	E

E:exist

NE:not exist

,where u is the x -coordinate of points with order 4 of which double points are both $(0, 0)$.

Proof. By Proposition 3, in any case that $A^2 - 4$ is quadratic non-residue, the curve has exactly two points of order 4 of which x -coordinate is either 1 or -1 and of which double point is the point $(0, 0)$. According to the lemma we show below, all we have to do to determine whether points of order 8 exist or not is to check that both $A + 2$ and $1/B$ are quadratic residue if the x -coordinate u is equal to 1, and both $-(A-2)$ and $-1/B$ are quadratic residue if the x -coordinate u is equal to -1. \square

Proposition 4. Let r be quadratic non-residue.

$$8|\#E^M \Leftrightarrow 8|\#E_r^M \text{ if } p \equiv 1 \pmod{4}$$

$$8|\#E^M \Leftrightarrow 8|\#E_r^M \text{ if } p \equiv 3 \pmod{4}$$

Proof. It is clear from the equation $\#E^M + \#E_r^M = 2(p+1)$ and Corollary 1. \square

To complete the proof of Theorem 2, we show the next lemma.

Lemma 1. Let $E^M : BY^2 = X^3 + AX^2 + X$ be a Montgomery-form elliptic curve. Then, both $u^2 + Au + 1$ and u/B are quadratic residue if a point (u, v) on E^M is the double point of some point on E^M . Conversely, in the case that $A^2 - 4$ is quadratic non-residue, a point (u, v) on E^M is the double point of some point on E^M if both $u^2 + Au + 1$ and u/B are quadratic residue.

Proof. Assume that the point (u, v) is the double point of some \mathbf{F}_p -rational point (x, y) on E^M . The formula of the tangent line at the point (x, y) is

$$Y = \frac{3x^2 + 2Ax + 1}{2By}(X - x) + y. \quad (4)$$

Since the tangent line (4) intersects the curve at the point $(u, -v)$, by substituting the point $(u, -v)$ for the pair of variables (X, Y) followed by multiplying $2By$ and squaring the both sides, we find the equation

$$4Bv^2By^2 = ((3x^2 + 2Ax + 1)(u - x) + 2By^2)^2. \quad (5)$$

Since the points (x, y) and $(u, -v)$ are on the curve, they satisfy the equations $By^2 = x^3 + Ax^2 + x$ and $Bv^2 = u^3 + Au^2 + u$. By using these equations, we find the equation

$$(3x^2 + 2Ax + 1)^2 - 4(x^3 + Ax^2 + x)(u + A + 2x) = 0. \quad (6)$$

Since $x \neq 0$, we divide the equation (6) by x^2 , and regard it as an equation with respect to $(x + \frac{1}{x})$. We find the equation

$$\left(x + \frac{1}{x}\right)^2 - 4u\left(x + \frac{1}{x}\right) - 4(Au + 1) = 0. \quad (7)$$

$x + \frac{1}{x} \in \mathbf{F}_p$ requires that the discriminant $4(u^2 + Au + 1)$ of the equation (7) should be quadratic residue. Thus, the number

$$u^2 + Au + 1 \quad (8)$$

should be quadratic residue. Let w be one of the square roots of $u^2 + Au + 1$. Then, the solutions of the equation (7) are $x + \frac{1}{x} = 2(u \pm w)$. $x \in \mathbf{F}_p$ requires that the discriminant of this equation with respect to the variable x

$$(u \pm w)^2 - 1 \quad (9)$$

should be quadratic residue. Thus, either $(u + w)^2 - 1$ or $(u - w)^2 - 1$ is quadratic residue, and the solutions of these equations are

$$x = (u + w) \pm \sqrt{(u + w)^2 - 1} \quad (10)$$

or

$$x = (u - w) \pm \sqrt{(u - w)^2 - 1}, \quad (11)$$

respectively. For simplicity, we set $\delta = u \pm w$. Then, we find the equation

$$By^2 = (2\delta + A) \left(\delta \pm \sqrt{\delta^2 - 1} \right)^2. \quad (12)$$

Thus, $y \in \mathbf{F}_p$ requires that the number

$$(2\delta + A)/B \quad (13)$$

should be quadratic residue. Hence, $(2(u + w) + A)/B$ is quadratic residue if $(u + w)^2 - 1$ is quadratic residue, and $(2(u - w) + A)/B$ is quadratic residue if $(u - w)^2 - 1$ is quadratic residue. From the equation

$$(u \pm w)^2 - 1 = u(2(u \pm w) + A), \quad (14)$$

we find the equation

$$((u \pm w)^2 - 1) \frac{2(u \pm w) + A}{B} = \frac{u}{B} (2(u \pm w) + A)^2. \quad (15)$$

Therefore, u/B is quadratic residue because the left-hand side of the equation (15) is quadratic residue.

Conversely, in the case that $A^2 - 4$ is quadratic residue, assume that both $u^2 + Au + 1$ and u/B are quadratic residue. Let w be one of the roots of $u^2 + Au + 1$. Let (x, y) be the point defined by (10), (11) or (12) depending on the conditions that both $(u + w)^2 - 1$ and $(2(u + w) + A)/B$ are quadratic residue, or both $(u - w)^2 - 1$ and $(2(u - w) + A)/B$ are quadratic residue. Then its double point is (u, v) . On the other hand, we have the following three equations.

$$((u + w)^2 - 1)((u - w)^2 - 1) = u^2(A^2 - 4) \quad (16)$$

$$(2(u + w) + A)(2(u - w) + A) = A^2 - 4 \quad (17)$$

$$((u + w)^2 - 1)(2(u + w) + A)/B = \frac{u}{B}(2(u + w) + A)^2 \quad (18)$$

These three equations (16), (17) and (18) lead that either both $(u + w)^2 - 1$ and $(2(u + w) + A)/B$, or both $(u - w)^2 - 1$ and $(2(u - w) + A)/B$ are quadratic residue. \square

6 Algorithms to Generate Elliptic Curves with Cofactor 4

In this section, we present an efficient algorithm for generating the Weierstrass-form elliptic curves with whose is equal to 4 and which have the Montgomery-form.

INPUT A prime $p (\geq 5)$.

OUTPUT A Weierstrass-form elliptic curve with the Montgomery-form and with cofactor 4.

1. Find r such that $(r/p) = -1$.
2. Generate a and b , and put $E : y^2 = x^3 + ax + b$.
3. Check the transformability to the Montgomery-form for E as follows.
 - 3.1 Check the equation $x^3 + ax + b = 0$ has a root in \mathbf{F}_p . Go to 2 if it has no roots.
 - 3.2 Check Condition 2 of Proposition 1 for any root of $x^3 + ax + b = 0$. Go to 2 if no roots satisfy the condition.

4. Check the divisibility by 8 for $\#E$ and $\#E_r$ by using Theorem 1. Go to 2 if they are divisible by 8.
5. Compute $\#E$ and check $\#E = 4l$ or $\#E_r = 4l$ for some prime l . Go to 2 if neither E nor E_r passes.
6. Check other security tests, and output the parameters of the curve if it passes all tests.

At Step 4, we can find the divisibility by 8 for the curve order by checking that just one of $A \pm 2$ and B is quadratic residue or not because we already know that $A^2 - 4$ is quadratic residue or not at Step 3.1.

In the case that $p \equiv 1 \pmod{4}$, if the equation has just one root at Step 3.1, we can find the transformability and the divisibility by checking $(3\alpha^2 + a)^{(p-1)/4} \equiv \pm 1 \pmod{p}$. If it is not equal to 1 or -1 , the curve is not transformable. If it is equal to 1, the curve is transformable and $8 \nmid \#E_r$, and if it is equal to -1 , the curve is transformable and $8 \nmid \#E$. In this case, we can remove both one computation time of square root and one computation time of quadratic residue.

In the case that $p \equiv 3 \pmod{4}$, we can discard the curve at Step 3.1 when the equation $x^3 + ax + b = 0$ has three roots in \mathbf{F}_p , because its curve order and its twist curve order are divisible by 8, if it is transformable to the Montgomery-form.

At Step 6, we use security tests like a MOV reduction ([MOV93]) to check whether the curve is suitable for cryptographic use ([FR94, MOV93, SA98, Sem98, Sma]).

We have implemented this algorithm, and have generated many Weierstrass-form elliptic curves with the Montgomery-form and with cofactor 4. The following curves are some of them. (See Appendix B for more numerical examples)

1. $\lceil \log_2 p \rceil = 160$

$$\begin{aligned} p &= f4a8058b \text{ eddbd6f3 } 9f656c5c \text{ 8c9f3244 } 9c4ae98b \\ a &= 771e67ee \text{ 7c7318f7 } c1b73997 \text{ f9f1794f } 2b80633c \\ b &= 60083263 \text{ 13ba95ec } 80bd966f \text{ 3d2752dd } 18c58c18 \\ \#E &= f4a8058b \text{ eddbd6f3 } 9f65d54c \text{ 4791a3bd } ffcb6f44 \\ &\quad = 3d2a0162 \text{ fb76f5bc } e7d97553 \text{ 11e468ef } 7ff2dbd1 \cdot 4 \\ A &= 082f1bf4 \text{ 912e93a6 } 7f283a64 \text{ e67eab15 } 15e34443 \\ B &= 8c26318c \text{ c1803eab } 069aaaff9 \text{ 882edbc9 } 0447d09d \\ \alpha &= af6c44a9 \text{ 93c02ad0 } 84ac19df \text{ 90a38a0a } 6ec8bca8 \end{aligned}$$
2. $\lceil \log_2 p \rceil = 192$

$$\begin{aligned} p &= 9ee8eff3 \text{ b36d910c } aec3c1ca \text{ 0e636af7 } c16db444 \text{ 5dee43a1 } \\ a &= 2e5453d8 \text{ bb581d59 } 5b937f50 \text{ 980f5344 } c698d336 \text{ 3983491d } \\ b &= 6f8bca6f \text{ 36dba7b7 } d5d5e9a2 \text{ 44c0bd43 } a0a8075d \text{ 8c3eb548 } \\ \#E &= 9ee8eff3 \text{ b36d910c } aec3c1ca \text{ f535369c } cc9c3692 \text{ c3245abc } \\ &\quad = 27ba3bfc \text{ ecdb6443 } 2bb0f072 \text{ bd4d4da7 } 33270da4 \text{ b0c916af } \cdot 4 \\ A &= 20f6fa01 \text{ d844b599 } b4f2e523 \text{ ea9bd066 } f8211bef \text{ c2eb9af0 } \\ B &= 56fa585e \text{ 5b366ebf } f680e2e5 \text{ cd2c5104 } 8e325147 \text{ 30fd2354 } \\ \alpha &= 783f4ef4 \text{ 1c25c7b9 } a52711ab \text{ 4c8a7f37 } a372dbe0 \text{ 3bec7feb } \end{aligned}$$

The Montgomery-form elliptic curves are not anomalous, since their curve orders are always divisible by 4 and are in the range $[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$.

We have already checked that discrete logarithm problems on the curves we have generated do not reduce to those on the extension fields of \mathbf{F}_p up to degree 512.

Remark 6. Since any Montgomery-form elliptic curve is transformable to the Weierstrass-form elliptic curve, the security of the Montgomery-form elliptic curve is identical to that of the Weierstrass-form elliptic curve. If there exists an efficient attack for the Montgomery-form, it is also efficient for the Weierstrass-form, and vice versa.

Since the best possible cofactor of the Montgomery-form elliptic curves is 4 and that of the Weierstrass-form elliptic curves is 1, the bit length of the base point orders of the Montgomery-form is shorter by two bits than that of Weierstrass-form on the same definition field.

Therefore, the security of the Montgomery-form for any attack except timing-attacks is slightly weaker (but no hindrances in cryptographic use) than or equal to that of the Weierstrass-form.

7 Extension Fields of \mathbf{F}_p

Using OEF(Optimal Extension Field) is a fast computation methods of the operations on the elliptic curves ([BP98, KMKH99]). Montgomery-form elliptic curves can be defined over the extension fields of \mathbf{F}_p as well as \mathbf{F}_p . Thus, Montgomery-form elliptic curves defined over OEF are attractive for speeding up the operations. In this section, we describe the results for elliptic curves defined over the extension fields of \mathbf{F}_p .

Let \mathbf{F}_{p^m} be the extension field of degree m .

Proposition 5. A Weierstrass-form elliptic curve $E/\mathbf{F}_{p^m} : y^2 = x^3 + ax + b$ defined over \mathbf{F}_{p^m} is transformable to the Montgomery-form elliptic curve $E^M/\mathbf{F}_{p^m} : BY^2 = X^3 + AX^2 + X$ defined over \mathbf{F}_{p^m} if and only if it satisfies two conditions as follows:

1. The equation $x^3 + ax + b = 0$ has at least one root in \mathbf{F}_{p^m}
2. The number $3\alpha^2 + a$ has quadratic roots in \mathbf{F}_{p^m} , where α is a root of $x^3 + ax + b = 0$ in \mathbf{F}_{p^m} .

Proposition 6. Let $r \in \mathbf{F}_{p^m}$ have no roots in \mathbf{F}_{p^m} , and let $E_r/\mathbf{F}_{p^m} : y^2 = x^3 + ar^2x + br^3$ be twist of $E/\mathbf{F}_{p^m} : y^2 = x^3 + ax + b$. Then, E is transformable to the Montgomery-form if and only if E_r is transformable to the Montgomery-form.

Proposition 7. Let $E^M/\mathbf{F}_{p^m} : BY^2 = X^3 + AX^2 + X$ be Montgomery-form elliptic curve. Both of $u^2 + Au + 1$ and u/B have quadratic roots in \mathbf{F}_{p^m} if (u, v) on E^M is the double point of some point on E^M . Conversely, in the case that $A^2 - 4$ has no quadratic roots in \mathbf{F}_{p^m} , (u, v) on E^M is the double point of some point on E^M if both of $u^2 + Au + 1$ and u/B have quadratic roots in \mathbf{F}_{p^m} .

Proof (of propositions). Substitute “have square roots in \mathbf{F}_{p^m} ” and “have no square roots in \mathbf{F}_{p^m} ” for “quadratic residue” and “quadratic non-residue”, respectively, in the proof of each proposition or lemma. \square

Therefore, we can obtain similar methods in \mathbf{F}_{p^m} by the propositions above.

8 Conclusion

In this paper, we show that the Montgomery-form elliptic curves are immune to the timing-attacks, and that the exact condition on the Weierstrass-form with/without the Montgomery-form. We also present an efficient algorithm for generating Weierstrass-form elliptic curves with Montgomery-form whose cofactor is exactly equal to 4. And this algorithm handles not only the original curve itself but also its twist so that it can find the good curve more efficiently. We also implement the algorithm and give some numerical examples obtained by this. In this paper, we should note that we mainly discuss elliptic curves over prime fields. However, the similar argument can be applied to any elliptic curves over any finite fields.

9 Acknowledgments

The authors would like to thank the anonymous referees for their helpful comments.

References

- [AMV93] Agnew,G.B., Mullin,R.C., Vanstone,S.A., *An Implementation of Elliptic Curve Cryptosystems Over $F_{2^{155}}$* , IEEE Journal on Selected Areas in Communications, vol.11,No.5, (1993), 804-813. [243](#), [255](#)
- [ANSI] ANSI X9.62, Public Key Cryptography for the Financial Services Industry, *The Elliptic Curve Digital Signature Algorithm(ECDSA)*, (1999). [239](#)
- [BP98] Bailey,D.V., Paar,C., *Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms*, Advances in Cryptology-CRYPTO'98,LNCS1462,(1998),472-485. [240](#), [253](#)
- [BSS99] Blake,I.F.,Seroussi,G.,Smart,N.P., *Elliptic Curves in Cryptography*, Cambridge University Press,(1999). [247](#)
- [CMO98] Cohen,H., Miyaji,A., Ono,T., *Efficient Elliptic Curve Exponentiation Using Mixed Coordinates*, Advances in Cryptology - ASIACRYPT '98, LNCS1514, (1998), 51-65. [242](#)
- [Cor99] Coron,J.S., *Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems*, Pre-Proceedings of Workshop on Cryptographic Hardware and Embedded Systems(CHES), (1999), 292-302. [244](#)
- [FR94] Frey,G., Rück,H.G., *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves*, Math. Comp. 62, (1994), 865-874. [252](#)
- [Izu99a] Izu,T., *Elliptic Curve Exponentiation for Cryptosystem*, SCIS'99,W4-1.1 (1999), 275-280. [242](#), [245](#)

- [Izu99b] Izu,T., *Elliptic Curve Exponentiation without y-coordinate*, Technical Report of IEICE. ISEC98-86 (1999), 93-98. 242, 245
- [KMKH99] Kobayashi,T., Morita,H., Kobayashi,K., Hoshino,F., *Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic*, Advances in Cryptology - EUROCRYPT'99, LNCS1592,(1999),176-189. 240, 253
- [Kobl87] Koblitz,N., *Elliptic curve cryptosystems*, Math. Comp.48, (1987), 203-209.
- [Koc] Kocher,C., *Cryptanalysis of Diffie-Hellman, RSA, DSS, and Other Systems Using Timing Attacks*, Available at <http://www.cryptography.com/> 239, 242
- [Koc96] Kocher,C., *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology - CRYPTO '96, LNCS1109, (1996), 104-113. 239, 242
- [Mil86] Miller,V.S., *Use of elliptic curves in cryptography*, Advances in Cryptology - CRYPTO '85, LNCS218,(1986), 417-426.
- [MOV93] Menezes,A., Okamoto,T., Vanstone,A., *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transaction on Information Theory, Vol. IT-39, No.5, (1993),1639-1646. 252
- [MOC98] Miyaji,A., Ono,T., Cohen,H., *Efficient elliptic curve exponentiation(II)*, SCIS'98,7.1.D,(1998)
- [Mon87] Montgomery,P.L., *Speeding the Pollard and Elliptic Curve Methods of Factorizations*, Math. Comp. 48, (1987), 243-264. 239, 241, 247
- [NIST99] National Institute for Standards and Technology, *Recommended Elliptic Curves for Federal Government Use*, (1999), Available at <http://csrc.nist.gov/encryption/> 239, 247
- [SA98] Satoh,T., Araki,K., *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*, Commentarii Mathematici Universitatis Sancti Pauli, (1998), 88-92. 252
- [OSK99] Ohgishi,K., Sakai,R., Kasahara,M., *Elliptic Curve Signature Scheme with No y Coordinate*, SCIS'99,W4-1.3 (1999), 285-287. 242
- [SEC-1] Standards for Efficient Cryptography, *Elliptic Curve Cryptography Ver.0.5*, (1999),Available at <http://www.secg.org/drafts.htm> 239, 247
- [Sem98] Semaev,I., *Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p*, Math. Comp. 67, (1998), 353-356. 252
- [Sma] Smart,N.P., *The Discrete Logarithm Problem on Elliptic Curves of Trace One*, to appear in Journal of Cryptology. 252
- [TK99] Takeuchi,K., Koyama,K., *Fast Computation of Elliptic Curve Cryptosystems*, SCIS'99,W4-1.2 (1999), 281-284 . 242

A Montgomery Scalar Multiplications on the Elliptic Curves Defined over the Finite Fields of Characteristic 2

The following is extracted from [AMV93].

Let E be an elliptic curve over \mathbf{F}_{2^m} having equation

$$y^2 + xy = x^2 + ax^2 + b$$

where $a, b \in \mathbf{F}_{2^m}, b \neq 0$. Let $P = (x_1, y_1, z_1)$ and $Q = (x_2, y_2, z_2)$ be two distinct and nonzero points on E with $P \neq -Q$. If $P + Q = (x_3, y_3, z_3)$, then

$$x_3 = AD \quad \text{and} \quad z_3 = A^3 z_1 z_2$$

where $A = x_2z_1 + x_1z_2$, $B = y_2z_1 + y_1z_2$, $C = A + B$ and $D = A^2(A + az_1z_2) + z_1z_2BC$. Since $-Q = (x_2, x_2 + y_2, z_2)$ and if $P - Q = (x_4, y_4, z_4)$, then

$$x_4 = A'D' \quad \text{and} \quad z_4 = (A')^3 z_1 z_2$$

where $A' = A$, $B' = x_2z_1 + B$, $C' = C + x_2z_1$ and $D' = D + z_1z_2[Bx_2z_1 + x_2z_1C + (x_2z_1)^2]$. Therefore,

$$x_4 = A[D + z_1z_2[Bx_2z_1 + x_2z_1C + (x_2z_1)^2]] \quad \text{and} \quad z_4 = A^3 z_1 z_2.$$

Thus, $x_3 = x_4 + z_1^2 z_2^2 x_1 x_2 A$ and $z_3 = z_4$.

It follows that to compute x_3 for $P + Q$, we need the x -coordinate of P, Q and $P - Q$. Now to compute kP , we compute $2P$ and then repeatedly compute $(2mP, (2m+1)P)$ or $((2m+1)P, (2m+2)P)$ from $(mP, (m+1)P)$, depending on whether the corresponding bit in the binary representation of k is a 0 or a 1. Since the difference in each pair is P , if we take the z -coordinate of P to be 1, then the z -coordinate of $(2m+1)P$ will always be 1. Hence, we can assume that either $z_1 = 1$ or $z_2 = 1$ in the formula for x_3 .

In the above, we may not assume the assumption that the z -coordinate of $(2m+1)P$ is 1 for any m , even though we take the z -coordinate of P to be 1. In this section, we clear that.

For avoiding the collision of the notation, we set $P = (x_P, y_P, z_P)$ which is used in “ kP ”. We substitute P and Q for $(m+1)P$ and mP , respectively, at the equation $P - Q = (x_4, y_4, z_4)$ above. Since the difference between $(m+1)P$ and mP is P , the point P is equal to the point (x_4, y_4, z_4) as a point, and *there exists some $\lambda \neq 0$ such that $(x_4, y_4, z_4) = (\lambda x_P, \lambda y_P, \lambda z_P)$ as a triple.* ((x_4, y_4, z_4) is the consequence of the subtraction using the method above.) Thus, if we assume $z_P = 1$, we find $z_3 = z_4 = \lambda$, and it is not always equal to 1.

Remark 7. The computing method such as Montgomery scalar multiplications without assuming either $z_1 = 1$ or $z_2 = 1$ in the formula for x_3 works and prevents timing-attacks. However, it is not efficiently fast. For a fast computation, we should combine the following relation between $P + Q$ and $P - Q$ with the formula above:

$$x_3 + x_4 = \frac{x_1 x_2}{(x_1 + x_2)^2},$$

where $P = (x_1, y_1), Q = (x_2, y_2), P+Q = (x_3, y_3)$ and $P-Q = (x_4, y_4)$ are points in affine coordinates. We omit the detailed descriptions for the fast computation.

B Numerical Examples

1. $\lceil \log_2 p \rceil = 224$

```

p = d0e7f3fc 9ed2398a 14ae970b db7b3d22 deb7715c 4ac259ca
    2c9ba8c3
a = 43ffe524 e92c14e8 c730f6cb e9dae99f 3bd1509b bcdd17bf
    330c1ca1
b = 07023dff eae7799e cea4c0ac a19b24fd 4ed0011f 4c7df255
    a9c22143
#E = 3439fcff 27b48e62 852ba5c2 f6def716 b3f4467d 0a57b6d8
    965f69df · 4
A = c7c856c0 8e60a802 45305c51 d49bfee1 fd5bfa7d 3a314a69
    f0f978e1
B = 8610c7be cb6d5f24 d10c6849 eb772f2b 181f4b05 2777d7fa
    a0828206
α = a6b6fcc2 51d65ccf 2c87630b c24f2827 a289a840 edfbe70b
    ce27ff2b

```

2. $\lceil \log_2 p \rceil = 256$

```

p =cff4508c b3e663a9 add65372 60ec1764 f633c64a da218c79
    e4f43d31 dd86b4f7
a = 303b6d25 e33dc651 edd322da 06b47d5c 1d57268bdbe0b152
    c1ae7731 4d8be56d
b = 5487b25a f80dcc71 428cee96 008dcdae 60ef4183 b8a91716
    9b6110d5 c9a4016c
#E = 33fd1423 2cf998ea 6b7594dc 983b05d9 4a5404fa 3332b622
    6ebabe0a 267c26fb · 4
A = 89f2a557 a80e151b 71963690 bf40a5e0 047c6d54 1f535115
    93e11b1c 99bcec6d
B = b69ff084 e42feff3 2f22b6cf 27ff2443 5d755e5a f4ca7f40
    c53a70a4 afaa1953
α = 474ff280 cfa17e98 9f39d4a0 9e9119a9 b39606f2 fcbbc6b22
    3f260ca3 8ceb0291

```

The following elliptic curve defined over \mathbf{F}_p where $\lceil \log_2 p \rceil = 162$ has the base point order size 160. This is equal to the base point order size of the elliptic curve with cofactor 1 defined over $\mathbf{F}_{p'}$ where $\lceil \log_2 p' \rceil = 160$.

3. $\lceil \log_2 p \rceil = 162$

```

p = 00000003 f224b887 e3fc28b7 f9a06aed f5da889e 032b3e37
a = 00000002 f700a850 72e6e12e dd8494c7 9ac083c2 a4bec8e0
b = 00000000 a1ad176a bb498420 27ac4b16 7ddd377d 6d2f8f02
#E = 00000000 fc892e21 f8ff0a2d fe687e72 84574b83 f79c0b73 · 4
A = 00000002 4852eaee 28edc219 6f3c9b3e 86f00972 1fa895af
B = 00000000 8d997623 607ebadb bffd2d7c3 9ee19a16 50f63e64
α = 00000000 271cf03a 8cb9c19f 82fb9840 5fe9e698 458750b7

```

Certificates of Recoverability with Scalable Recovery Agent Security

Eric R. Verheul

PricewaterhouseCoopers, GRMS Crypto Group, P.O. Box 85096, 3508 AB Utrecht
Eric.Verheul@[nl.pwctglobal.com, pobox.com]

Abstract. We propose new schemes for Certificates of Recoverability (CRs). These consist of a user's public key and attributes, its private key encrypted in such a way that it is recoverable by one or more Key Recovery Agents (KRAs), plus a publicly verifiable proof of this (the actual CR). In the original schemes, the level of cryptographic security employed by the KRA and the users is necessarily the same. In our schemes the level of cryptographic security employed by the KRA can be set higher, in a scalable fashion, than that being employed by the users. Among the other improvements of our schemes are its applicability to create CRs for cryptosystems based on the Discrete Log problem in small subgroups, most notably the Digital Signature Standard and Elliptic Curve Crypto systems. Also, the size of the constructed proofs of knowledge can be taken smaller than in the original schemes. We additionally show several ways to support secret sharing in our scheme. Finally we present several new constructions and results on the hardness of "small parts", in the setting of Diffie-Hellman keys in extension fields.

1 Introduction

In a Public Key Infrastructure there are several roles, of which we name:

Participants Who rely on public key encryption and need to be confident that the associated private key is owned by the user mentioned in the public key.

Certification Authorities This confidence is obtained through the use of *public key certificates*, which bind public keys and its attributes to a user. The binding is achieved by having a trusted *Certification Authority* verifying the user's identity and other attributes and then digitally sign each certificate. This verification is usually performed by an optional entity, called *Registration Agent* "closer to the user" than the CA.

Key Recovery Agents Sometimes a user or the organization the user works for, requires an emergency back-up of its private key and deposits it with a trusted Key Recovery Agent (KRA).

It could be part of a Certification Policy of a user's organization or CA that no certificates are issued unless the associated private key is properly escrowed with a KRA. Of course, the KRA should operate under a strict, well documented, and

regularly audited, recovery policy. In [27] the concept of *Certificate of Recoverability* (CR) is introduced which supports such policies. Here a user offers three components to a CA (or RA):

- C1** Its public key, identity and other attributes.
- C2** An encrypted copy of its private key, decryptable only by one or more Key Recovery Agents.
- C3** A publicly verifiable proof that the first two components are correctly formed, i.e. that the private key of the public key in component **C1** coincides with the encrypted private key in component **C2**. This component is called the Certificate of Recoverability.

After verifying the proof in **C3**, the CA issues a regular certificate on the public key. In addition, the CA archives the three components or sends them to the KRA(s). In a conventional escrow scheme, i.e. without the component **C3**, the CA (or RA) would have setup a KRA-acceptance phase. This consists of sending components **C1** and **C2** to all the KRAs to verify the validity of **C2**. The central advantage of a CR scheme compared with a conventional escrow system, is that the CA can perform this KRA-acceptance phase without having to communicate with the KRA(s). This gives rise to the many advantages of a CR scheme, like speed, cost effectiveness and security.

The publicly verifiable proof in component **C3** should preferably have the property of being perfectly zero-knowledge, or being a parallel version of it, see [26]. Also, observe that the user's private key is contained in both his public key and in component **C2**. So, preferably there should be a guarantee for *combined* security, i.e. that this combination of encryption schemes (used by users and KRAs) does not result in an exploit, yielding (parts of) the user's private key. Compare [11], where combined (in)security is discussed for the RSA cryptosystem.

In [27] a construction for the CR concept is given, which is related to a construction in [24]. Here a prime number p and a cyclic group $G = \langle g \rangle$ of order p are generated (for instance by the CA) in which the discrete logarithm problem is intractable. The idea is that the public keys of users take the form g^x , where $0 < x < p$, is the associated private key.

In addition, the ElGamal encryption scheme [8] in a multiplicative subgroup Γ of order ω in the (basic) finite field $GF(p)$ is used to encrypt a copy of the private key x in **C2**. The last component **C3** consists of a transcript of a perfect zero-knowledge proof of knowledge, proving that the private key x is encrypted in both **C1** and **C2**. Moreover, a guarantee of combined security for this construction is given in [24].

In both [24] and [27] it is suggested to combine the construction of the system parameters of users and KRAs. One starts with choosing the order ω as a large prime number, such that $p = 2\omega + 1$ and $r = 2p + 1$ are prime as well. Then take G as the subgroup of order p in the multiplicative group of $GF(r)$ and Γ is the subgroup of order ω in the multiplicative group of $GF(p)$.

For sufficient security, r, p, ω should be of sufficient size, of say ≥ 1000 bits. Also, to obtain a soundness level of 2^{-v} in the above scheme, the size of a transcript is roughly v times the number of bits in p , e.g. 100,000 bits for $v = 100$ and $|p| = 1000$.

This type of construction of CRs has two major drawbacks:

1. The level of security given by the KRA's public key is the same as the security given by the public keys of the users. The KRA's private key gives access to *all* users' private keys, as it is a *master key*. Hence the KRA's public key is subject to more severe threats than the users' public keys. For instance, attackers wanting access to communication of different users, could form an alliance to break the KRA's public key. It is "best practice" that master keys, such as KRA's public keys, should be considerably more difficult to break than user's keys, i.e. public keys of the users. This is also why the signing key of a (root) Certifying Authority is typically longer than that of its users.

We remark that using stronger KRA's public keys is only sensible, provided that breaking *one* user's public key does not imply breaking *all* users' public keys. This is the case when using DL-based schemes. In the setting of a finite field, breaking one discrete logarithm based public key considerably helps in breaking other ones, but that still takes sub-exponential time (cf. [10], [26, p.172]). In the setting of Elliptic Curve Cryptosystems, breaking one public key doesn't help (much) in breaking other public keys: an exhaustive search proportional to the square root of the group's order still has to be conducted.

2. The use of DL-based schemes in groups of relative small order (of say of size 160 bit), can be very beneficiary in terms of speed and data-storage. Such groups are employed in the Digital Signature Standard and in Elliptic Curve Crypto systems. However, the schemes from [27], forces users to use groups of orders of at least 1024 bits to have sufficient security; making it impossible to employ these beneficiary groups.

To remedy these two drawbacks, one could use the generic verifiable encryption scheme that appears in [2]. However, the resulting interactive proof in **C3** would only have the property of computational zero-knowledge. Also, no formal guarantee for *combined* security, as defined above, is given in [2].

Our System We propose new schemes for CRs which do not have the above mentioned drawbacks. In our schemes, users can use any type of cyclic group G of prime order p in which the discrete logarithm is intractable. This means that the size of p should be sufficiently large, of say ≥ 160 bits; but no further restrictions are made. The user's private key is encrypted for the KRA, using the ElGamal scheme in a field extension $GF(p^t)$. Here one can choose to employ the classical ElGamal scheme, i.e. using the whole multiplicative group of $GF(p^t)$, or the subgroup variant as proposed by Schnorr [23], using a multiplicative subgroup of $GF(p^t)$ of suitable, large prime order.

In both variants the proofs of knowledge, **C3**, can be formed in a perfect zero-knowledge fashion, the size of which can be taken considerably smaller than that

from [27] and [24]. Moreover, by employing optimal normal bases our schemes are more efficient than comparable schemes in [27]. Finally, the guarantee for combined security is proven to be given by the hardness of the Diffie-Hellman Decision Problem.

Apart from key-escrow applications, our techniques can be applied in the design of electronic cash systems providing revocable anonymity [25] in which the identity of the system's users can be recovered when the system is abused for criminal activities. The recovery of the identity is done with the help of trustees, whose role can be compared with that of the above mentioned KRAs.

We remark that we do not incorporate the CR (i.e. **C3**) itself in the issued certificate. There is no need to do so, and in our schemes this would also introduce the possibility of the establishment of shadow public keys, see [12].

Outline of the paper Apart from presenting new schemes for Certificates of Recoverability, we present several new results and constructions in multiplicative subgroups of extension fields. In Section 2 we will first discuss new results on the hardness of “small parts” of the key exchanged in the Diffie-Hellman protocol in extension fields. Section 3 deals with proving knowledge on equality of logarithms in the setting of extension fields. In Section 4, we present our basic scheme, using the results of Sections 2 and 3 to prove security and functionality. In this section we also describe generalizations, like incorporating secret sharing, and efficiency improvements of the scheme.

2 DL-Based Cryptosystems over Extension Fields

2.1 Introduction

Let H be a multiplicative, cyclic group in which the discrete logarithm problem is intractable and h is a generator of H of order o . The h, H and o are system parameters given to all participants. The Diffie-Hellman (DH) key agreement [7] was the first practical solution for allowing two parties to agree over an insecure channel on a common secret key. The security of it lies in the *Diffie-Hellman problem* of computing values of the function $\text{DH}(h^x, h^y) = h^{xy}$. A drawback of the basic DH scheme is that the parties involved, Alice and Bob say, can only share one secret key. This is remedied in several schemes related to the DH scheme, such as the ElGamal encryption scheme [8]. Here Bob picks a random $0 < x < o$ and publishes his public key $y = h^x$. If Alice wants to send a message $m \in G$ to Bob, then Alice picks a random $0 < k < o$ and sends $(A, B) = (h^k, m \cdot y^k)$ to Bob. Upon receipt, Bob decrypts the message by forming B/A^x which is equal to the message m .

Two other problems are related to DH problem. The first one is the *Diffie-Hellman Decision* problem: given $a, b, c \in \langle h \rangle$ determine whether $c = \text{DH}(a, b)$. The DH problem is at least as difficult as the DH Decision problem. The second related problem is the *Discrete Logarithm* (DL) problem in $\langle h \rangle$ is given $a \in \langle h \rangle$, find $0 < x < o$ such that $a = h^x$. The DL problem is at least as difficult as the

DH problem. For the security of the Diffie-Hellman scheme, all three problems should be intractable.

We call h' a *prime subgenerator* of h if $h' \in \langle h \rangle$ and the order of h' is a prime number. By virtue of the Pohlig-Hellman algorithm [17], the difficulty of the DL problem w.r.t. h , given the factorization of the order of h , is as difficult as solving the DL problem for all prime subgenerators of h .

Clearly, one can construct the Diffie-Hellman and related schemes in any multiplicative subgroup $\Gamma = \langle \gamma \rangle$ of $GF(p^t)$ of order ω . For solving the discrete logarithm problem for a prime subgenerator γ' of γ of order ω' , one can use an index calculus (IC) based algorithm that has a heuristic expected asymptotic running time of $L(p^s, 1/3, 1.923 + o(1))$, see [1] and [13], where s is the smallest divisor of t such that $\langle \gamma' \rangle$ is contained in a subfield of $GF(p^t)$ isomorphic to $GF(p^s)$. If $p = 2$ then the constant 1.923 can be replaced by 1.587, see [6]. Alternatively one can use Birthday Paradox (BP) based algorithms (e.g. Pollard's rho algorithm [18]) that have expected running times exponential in the size of the ω' . More precisely, breaking the Discrete Logarithm problem can be solved in expected $O(\sqrt{\omega'})$ elementary operations in $GF(p^t)$.

This leads us to the conclusion from [13] that *w.r.t. attacks known today* the intractability of the discrete logarithm problem in Γ depends on the existence of a prime generator γ' of γ whose minimal surrounding subfield and prime order are of sufficient size. The particular form of the field itself, is not relevant. In other words, if $GF(p^t)$ is the minimal surrounding field of a subgroup of prime order, then - *w.r.t. attack known today* - the discrete logarithm in this subgroup is approximately as difficult as the discrete logarithm in a subgroup of prime order of a basic field $GF(P)$ if the size of P is approximately equal to as t times the size of p , and the order of both subgroups are about the same size.

Hence, a suitable generator γ in a field extension $GF(p^t)$ should have a prime subgenerator that is not contained in one of the proper subfields and should have a suitably large order. In [13], A.K Lenstra proposes a simple, practical method for the construction of a field extension and suitable generator generating a relatively small subgroup in it. The idea is that one fixes the size of the prime number p and a number t such that p^t is “large” enough. Then one looks for a large prime factor ϖ in the value of the cyclotomic polynomial $\phi_t(p)$. The latter can be done using trial divisions with the primes up to, say 10^5 ; any other reasonable bound or method will do. Finally, one constructs a generator γ of order ϖ , by looking for an element different from 1, such that $g^{(p^t-1)/\varpi} = 1$. If one factorizes $p^t - 1$, one can also find a generator of the whole multiplicative group of $GF(p^t)$.

Using the above construction, the size of ϖ is about $\varphi(t) \cdot |p|$ bits (where $\varphi(\cdot)$ is Euler's totient function) which grows as least as fast as $p^{t/\log(\log(t))}$. The complexity of the BP based algorithms grows much faster, than the complexity of the IC based algorithms. So if the size of the surrounding field is large enough to resist the sub-exponential, IC based algorithms, then the order ϖ will usually be large enough “automatically” to resist the BP based algorithms as well.

2.2 Partial Security of DL-Based Systems over Extension Fields

Let $\gamma \in GF(p^t)$ be a generator of a group Γ of order ω . In our applications, γ is not contained in a genuine subfield of $GF(p^t)$ and ω is either equal to a large prime number ϖ or to $p^t - 1$. We start with some well-known notions; if J is an element of $GF(p)[X]$, i.e. a polynomial with coefficients in the basis field $GF(p)$, then for any natural number i , the i -th coefficient of J is denoted by $[J]_i$. If $F = \sum_{i=0}^t a_i X^i$ is an irreducible polynomial of degree t in $GF(p)[X]$, then we can describe the extension field $GF(p^t)$ as $GF(p)[X]/(F)$, i.e. each element f in $GF(p^t)$ can be uniquely written modulo F , as a polynomial of degree $< t$. In this setting, for any natural number i less than the degree of F , we let the i -th coefficient $[f]_i$ denote $[f \bmod F]_i$.

There are many such representations and some have special properties. For instance, if $t + 1$ is also prime and that p is a primitive element modulo $t + 1$, then one can use a special irreducible polynomial, the zeros of which form an optimal normal basis. Which such a basis, one can do exponentiation in $GF(p^t)$ even more efficiently than in a basic field of comparable size. See [13], where it is also shown how such pairs p, t , and the special irreducible polynomial can be easily generated.

Some constructions, like the coefficients, are dependent of the concrete representation of $GF(p^t)$ one has chosen. Throughout the remainder of this paper we assume that one has chosen an irreducible polynomial F , yielding a concrete representation of $GF(p^t)$.

Below we prove that computing coefficients (i.e. “small parts”) of the Diffie-Hellman key, is as difficult as computing the whole key. These results are similar to the ones in [4] where the security of the Most Significant Bits (MSB) of Diffie-Hellman keys in a basic finite field $GF(p)$ is studied.

Theorem 21 *We use the above terminology. Given an oracle that computes a coefficient of a Diffie-Hellman key γ^{xy} on basis of γ^x , γ^y , then there exists a polynomial time algorithm that solves the Diffie-Hellman problem in $\Gamma = \langle \gamma \rangle$.*

Proof: The function that returns a coefficient of a fixed position less than t is linear and is hence a summing function (see below) by Proposition 23 below. The result now immediately follows from Theorem 24 below. \square

It follows in particular that *any* bit of the Diffie-Hellman key in $GF(2^t)$ is as hard to compute as the whole. This offers a more conventional alternative for the new variant of the Diffie-Hellman protocol mentioned in [4] having exactly this property. In our situation, the size of p is not small (≥ 160 bit or more) and we immediately obtain the following corollary, which is crucial for the security of our scheme. This result immediately follows directly from Theorem 21.

Corollary 22 *Form, in the above terminology, the public key $\psi = \gamma^x$ with private key $0 < x < \omega$. Then solving $S \in GF(p)$ from the ElGamal encryption $(\gamma^k, S \cdot [\psi^k]_i)$ for some fixed $0 \leq i \leq t - 1$ and $1 \leq k \leq \omega$ random, is equivalent to solving the Diffie-Hellman problem in $\Gamma = \langle \gamma \rangle$.*

Theorem 21 is a consequence of a much broader result. Let n be a non-negative number and consider the integers e_1, \dots, e_n (the “exponents”) and the elements $\lambda_1, \dots, \lambda_n \in GF(p^t) \setminus \{0\}$ (the “multipliers”) and consider the following *summing* function $Z(\cdot) : \langle \gamma \rangle \rightarrow GF(p^t)$ defined by:

$$Z(\kappa) = \sum_{i=1}^n \lambda_i \cdot \kappa^{e_i}.$$

The number n is called the *degree* of the summing function and the number $d = \gcd(e_1, e_2, \dots, e_n, \text{ord}(\gamma))$ is called the *order* of the summing function. Note that if all multipliers are elements of the basic field $GF(p)$, then its form is representation-independent.

The following result implies that the collection of summing functions is rather large.

Proposition 23 *If $f(\cdot)$ is a linear mapping from $GF(p^t)$ onto $GF(p)$ (i.e. a functional), then the restriction of $f(\cdot)$ to $\langle \gamma \rangle$ is a summing function of order 1. Moreover, the multipliers occurring in the definition of summing function can be easily determined. In particular it follows that the restriction to $\langle \gamma \rangle$ of the trace function $Tr(\cdot)$ of $GF(p^t)$ onto the base subfield $GF(p)$ defined by (cf. [16]):*

$$Tr(\alpha) = \alpha + \alpha^p + \dots + \alpha^{(p^{t-1})}.$$

is a summing function of order 1.

Proof: It is evident that the trace function is a summing function of order 1. From [16, Theorem 2.24] it follows that (the restriction to $\langle \gamma \rangle$ of) any functional is of the form $\alpha \rightarrow Tr_K(\beta \cdot \alpha)$ for some fixed $\beta \in GF(p^t)$ and is hence a summing function of order 1.

We note that given a functional $f(\cdot)$, of which we assume it can be efficiently evaluate in our concrete representation, one can easily compute the associated β (and thus the multipliers) from $f(1), f(\gamma), \dots, f(\gamma^{t-1})$ using the following equality:

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ \gamma & \gamma^p & \dots & \gamma^{p^{t-1}} \\ \vdots & \vdots & \dots & \vdots \\ \gamma^{t-1} & \gamma^{(t-1)p} & \dots & \gamma^{(t-1)p^{t-1}} \end{pmatrix} \cdot \begin{pmatrix} \beta \\ \beta^p \\ \vdots \\ \beta^{p^{t-1}} \end{pmatrix} = \begin{pmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{t-1}) \end{pmatrix} \quad (1)$$

Recall that γ is an element in $GF(p^t)$ that can not be embedded in a proper subfield of $GF(p^t)$, so $\gamma, \gamma^p, \dots, \gamma^{p^{t-1}}$ are all distinct and the above matrix is a non-singular matrix of the Vandermonde type. \square

Theorem 24 *In the above terminology, let $Z(\cdot)$ be a summing function of order d . Also let \mathcal{O} be an oracle that on basis of any γ^a and γ^b computes $Z(\gamma^{ab})$. Then there exists a polynomial time algorithm that computes γ^{abd} on basis of γ^a and γ^b . That is, for $d = 1$ there exists a polynomial time algorithm that solves the whole Diffie-Hellman problem in $\langle \gamma \rangle$.*

Proof: Let $V = \gamma^x$ and $W = \gamma^y$ be any elements of $\langle \gamma \rangle$. With the help of oracle \mathcal{O} one can not only determine $Z(\gamma^{xy})$, but also $Z(\gamma^{x(y+i)})$ (by giving the oracle the input V and $\gamma^i \cdot W$) for $i = 0, \dots, n - 1$, where n denotes the order of $Z(\cdot)$. This results in the following equality:

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ V^{e_1} & V^{e_2} & \dots & V^{e_n} \\ \vdots & \vdots & \dots & \vdots \\ V^{(n-1) \cdot e_1} & V^{(n-1) \cdot e_2} & \dots & V^{(n-1) \cdot e_n} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \cdot \gamma^{xye_1} \\ \lambda_2 \cdot \gamma^{xye_2} \\ \vdots \\ \lambda_n \cdot \gamma^{xye_n} \end{pmatrix} = \begin{pmatrix} Z(\gamma^{xy}) \\ Z(\gamma^{x(y+1)}) \\ \vdots \\ Z(\gamma^{x(y+n-1)}) \end{pmatrix} \quad (2)$$

The above matrix is of the Vandermonde type. Let us first consider the case that all $V^{e_1}, V^{e_2}, \dots, V^{e_n}$ are all different. Then this matrix is regular, and so from equality (2) one can determine the elements $\gamma^{xy \cdot e_1}, \gamma^{xy \cdot e_2}, \dots, \gamma^{xy \cdot e_n}$. From this one can determine the element γ^{xyd} by taking a suitable combination of the $\gamma^{xy \cdot e_1}, \gamma^{xy \cdot e_2}, \dots, \gamma^{xy \cdot e_n}$.

To prove the general case, observe that if V^{e_i} and V^{e_j} are equal then so are γ^{xye_i} and γ^{xye_j} . So by restricting to a maximal collection of different V^{e_i} , one can determine γ^{xyd} by the above argument. \square

We again use the above terminology. An element $\omega = \gamma^x \in \langle \gamma \rangle$ can also be described in terms of its minimal polynomial of degree t . This representation is not unique, as all conjugates of ω , i.e. $\omega, \omega^p, \omega^{p^2}, \dots, \omega^{p^{t-1}}$, have the same minimal polynomial. The i -th order coefficient of this polynomial equals

$$(-1)^i S_i(\omega, \omega^p, \omega^{p^2}, \dots, \omega^{p^{t-1}}), \quad (3)$$

where S_i denotes the elementary symmetric polynomial in t variables of degree i . One can easily prove that expression (3) yields a summing function. By virtue of [13, Lemma 2.4] it follows that for $1 \leq i \leq t - 1$, the order of this summing function can not contain prime factors $\geq i$. That is, the order of this summing function is very small. Hence it follows from Theorem 24 that for $1 \leq i \leq t - 1$, determining an i -th order coefficient of the minimal polynomial of the Diffie-Hellman key, is as difficult as determining the whole key.

Note that one can easily determine the minimal polynomial of γ^{xy} on basis of the minimal polynomial of either γ^x and y or the minimal polynomial of γ^y and x . On this idea variants of the Diffie-Hellman scheme can be based in which the involved parties send each other the minimal polynomials γ^x, γ^y rather than the elements themselves. The exchanged secret is the minimal polynomial of the element γ^{xy} , or some of the non-zero order coefficients of this. More generally one can take the value of a summing function of low order, evaluated in γ^{xy} .

It easily follows from Theorem 24 that such variants are as at least as secure as the original Diffie-Hellman scheme in our concrete representation of $\langle \gamma \rangle$. As minimal polynomials are representation-independent it directly follows that breaking such a variant means breaking the original Diffie-Hellman scheme in any representation of $\langle \gamma \rangle$. The converse, which is less interesting for the security of our variant, is also true but its proof is not elementary. The proof follows by determining zeros of the exchanged minimal polynomials in a fixed representation,

which can be done in deterministic polynomial time by the technique of H.W. Lenstra Jr. in [15].

The above idea is used in [5] to construct variants of the Diffie-Hellman scheme in which the number of bits exchanged is only a third of what is normally used, while the offered security against attacks known is the same.

Thus far, we have only discussed the unlikeliness that non-noisy oracles exist, that always give correct answers. Now we will briefly discuss the possible existence of noisy oracles \mathcal{O} that on basis of uniformly random γ^x and γ^y in $\langle \gamma \rangle$ compute $Z(\gamma^{xy})$, with non-negligible probability ϵ . In the general context *non-negligible* means that $1/\epsilon$ is less than a polynomial in $\log_2(|GF(p^t)|) = t \cdot \log_2(p)$. A first indication that such oracles do not exist follows from the easily verified observation that with such an oracle one can solve the Diffie-Hellman Decision problem in $\langle \gamma \rangle$, that is considered hard. See [19].

In our context (the scheme in section 4) it's more natural to say that a probability ϵ is *non-negligible* if $1/\epsilon$ is less than a polynomial in $\log_2(p)$. Indeed, a (small) parameter t is designed and then fixed; only the parameter p then varies. We have a heuristic proof for the following variant of Theorem 24 (of which similar variants of Proposition 21 and Corollary 22 can be easily deduced).

Theorem 25 *Let $Z(\cdot)$ be a summing function on Γ of degree 1. Given a (noisy) oracle \mathcal{O} that on basis of uniformly random γ^x and γ^y computes $Z(\gamma^{xy})$, with non-negligible probability ϵ then there exists a probabilistic polynomial time algorithm that solves the Diffie-Hellman problem in $\langle \gamma \rangle$.*

3 Double Deckers in Field Extensions

From now on we let $G = \langle g \rangle$ be a cyclic group of order prime order p in which the discrete problem is intractable. We introduce a new notion, that combines G , $GF(p)[X]$, and $GF(p^t)$.

For a polynomial J in $GF(p)[X]$ we define the *power* polynomial, g^J by

$$\sum_{i=0}^{\deg(J)} g^{[J]_i} X^i.$$

Similar to the convention used for regular polynomials, coefficients of powers that are not mentioned are assumed to be equal to the “zero element”, which is $g^0 = 1$. J is called the *exponent* polynomial of g^J . If $F = \sum_{i=0}^t a_i X^i$ is an (irreducible) polynomial of degree t in $GF(p)[X]$, then we obtain a natural equivalence on the power polynomials: two power polynomials are equivalent iff the difference of their exponent polynomials is a multiple of F ; the resulting equivalence group is called the group of *power polynomials modulo F*. It is clear that we can represent power polynomials modulo F as t tuples. The collection of power polynomials, resp. power polynomials modulo a polynomial F of degree t will be denoted by \mathcal{E} , resp. $\mathcal{E}/(F)$.

Let $P_1 = \sum_{i=0}^m U_i X^i$ and $P_2 = \sum_{i=0}^n V_i X^i$ be two power polynomials. Then the product of $P_1 \cdot P_2$ is defined as the power polynomial $\sum_{i=0}^{\max(m,n)} U_i \cdot V_i X^i$. The inverse P_1^{-1} of P_1 is defined as $\sum_{i=0}^m U_i^{-1} X^i$. If $P_1 = g^A$ and $P_2 = g^B$, then it follows that the $P_1 \cdot P_2 = g^{A+B}$ and $P_1^{-1} = g^{-A}$. Furthermore, if $C = \sum_{j=0}^r c_j X^j$ is a polynomial in $GF(p)[X]$, then P_1^C is defined as $\sum_{k=0}^{k=m+r} \prod_{i+j=k} U_i^{c_j} X^k$. It follows that if $P_1 = g^A$, then P_1^C is equal to $g^{A \cdot C}$.

Proposition 31 *Let $P_1 = g^A = \sum_{i=0}^m U_i X^i$ be a power polynomial of degree m , and let F be an (irreducible) polynomial of degree t in $GF(p)[X]$. Then (without knowing A) one can determine $g^{A \bmod F}$.*

Proof: For a proof, let $F = \sum_{i=0}^t f_i X^i$, with $f_t \neq 0$. Observe that

$$g^{(A - \frac{a_m}{f_t}) \cdot F \cdot X^{m-t}} = g^A \cdot (U_m)^{-f_t^{-1} F \cdot X^{m-t}}.$$

which is a power polynomial of degree $< m$, equivalent with P_1 modulo F and constructible without knowledge of A . The proposition now follows from an inductive argument. \square

As we remarked before, we represent any p -ary polynomial modulo an irreducible p -ary polynomial F of degree t (i.e. an element in $GF(p^t)$) as a p -ary polynomial of degree $< t$. We will also represent any p -ary power polynomial modulo F as one of degree $< t$. By the virtue of Proposition 31 we can determine this representation *without* explicitly knowing the exponent polynomial.

As before, we let γ be an element of a multiplicative group of $GF(p^t)$, that is not contained in a proper subfield of $GF(p^t)$ and let $\psi \in \Gamma = \langle \gamma \rangle$. Also, let the order of γ be denoted by ω .

Now, suppose that person P (for prover) gives the elements $\mu \in GF(p^t)$ and $M \in \mathcal{E}$ to person V (for verifier) and states:

Assertion DD: There exists a number k less than ω such that

$$\mu = \gamma^k \quad M = g^{(\psi^k)} \tag{4}$$

The following protocol, a variant of the double decker protocol in [24], lets P prove statement DD without revealing anything about k or ψ^k .

Protocol 32

Pr-1 *The Prover generates a random number l less than the order ω of γ , calculates $\nu = \gamma^l$, $N = g^{(\psi^l)}$ and hands ν and N over to V .*

Pr-2 *V generates a random $h \in \{0, 1\}$, and presents h as a challenge to P .*

Pr-3 *P calculates $z = l - h \cdot k \pmod{\omega}$ and hands z over to V .*

Pr-4 *V verifies that both*

Pr-4a. $\nu = \gamma^z \cdot \mu^h$, and

Pr-4b. $N = g^{(\psi^z)}$ if $h = 0$ and $N = M^{(\psi^z)}$ if $h = 1$.

The protocol satisfies the following properties which are easily verified.

Completeness If statement DD is true, then V will accept it.

Soundness If L-E is not true, then with a probability less than $1/2$ it will be accepted by V .

Security If L-E is true, then V can not learn any secret information on k or ψ^k by following the protocol.

Using the Fiat-Shamir heuristic [9] one can convert Protocol 32 in a non-interactive scheme with knowledge error 2^{-v} for a security parameter v . To this end, let $H(\cdot)$ be a secure hash function with output length equal to v bits. For $i = 1, \dots, v$ the Prover chooses $0 < l_i < \omega$ randomly and calculates (as in Pr-1 above) $\nu_i = \gamma^{l_i}$ and $N_i = g^{(\psi^{l_i})}$. Then he computes the v -tuple

$$Z = (z_1, \dots, z_v) = (l_1 - h_1 \cdot k \pmod{\omega}, \dots, l_v - h_v \cdot k \pmod{\omega})$$

where h_i denotes the i -th bit of $H = H(\mu, M, \nu_1, N_1, \dots, \nu_v, N_v)$. The non-interactive proof consists of Z and H . To verify, for $i = 1, \dots, v$, one re-constructs ν_i, N_i by:

$$\begin{aligned} \nu_i &= \gamma_i^z \cdot \mu^{h_i} \\ N_i &= \begin{cases} g^{(\psi^z)} & \text{if } h_i = 0 \\ M^{(\psi^z)} & \text{if } h_i = 1 \end{cases} \end{aligned}$$

and then verifies that $H(\mu, M, \nu_1, N_1, \dots, \nu_v, N_v)$ equals H . The size of the non-interactive proof is about $\omega \cdot v$ bits.

As is also explained in [3], a (dishonest) Prover could try to guess the v bits, h_i , and then form the accompanying responses z_i . If the Prover's ability to try this is bound by K trials, then the *probability of failure* of the non-interactive proof equals $K \cdot 2^{-v}$. The choice of v should take this into account. In [3] it is indicated that for a probability of failure less than 2^{-w} , one should take $v = 2w$, that is $K = 2^v$. However, this is a correct asymptotic choice and rather arbitrary in practice.

4 Our Scheme

4.1 The Basic Scheme

As before, let $G = \langle g \rangle$ be a multiplicative, cyclic group of prime order p in which the discrete logarithm problem is intractable. We will not further specify G , but in a typical example G is the multiplicative group of a finite field or the group of points on an elliptic curve over a finite field. The g, G and p are system parameters that are not yet fixed; only the size $|p|$ of p is fixed. As in the scheme of [27], we will generate g, G, p until we find one for which we can construct a suitable KRA encryption scheme (see below). The user's public key will be of the form $y = g^x$ where $0 < x < p$ is the user's private key.

Our security objective is to design an asymmetric cryptosystem based on the Discrete Logarithm problem in a finite field that has “asymmetric security” of

D bits, e.g. $D \geq 1024$. To this end, choose a number t such that $t \cdot |p| \geq D$. Next, construct random (user) system parameters g, G, p such that p is of size t . Next look for a large enough prime number ϖ in the value of the cyclotomic polynomial $\phi_t(p)$ (see Section 2), to support the security objective. If this fails, generate new random (user) system parameters g, G, p . Next, generate γ in $GF(p^t)$ of order ω either equal to $p^t - 1$ or ϖ . For the first generation, a factorization of $p^t - 1$ is required. So γ either generates the whole multiplicative group of $GF(p^t)$, or a subgroup of order ω . In either case, the Discrete Logarithm is infeasible w.r.t. attacks known today (see Section 2).

Both options correspond with the two types of ElGamal encryptions (classical and subgroup based) the KRA can use, each leading to a different scheme for CRs. In either case, the KRA's public key will be of the form $\psi = \gamma^{x_K}$ where $0 < x_K < \omega$ is the KRA's private key.

Both schemes actually support CRs for a variable number s (with $1 \leq s \leq t$, of different public/private keys $y_{s-1}, \dots, y_0 \in G$ owned by the same user (e.g. supporting different applications or Certification Policies) simultaneously. So, by taking $s = 1$ in the below schemes we get a Certificate of Recoverability as defined by Young and Yung, for higher s we get a multiple version of this.

Scheme 41 *The user generates his s private keys, x_0, \dots, x_{s-1} and forms the following three components:*

C1 *The user's public keys $y_{s-1} = g^{x_{s-1}}, y_{s-2} = g^{x_{s-2}}, \dots, y_1 = g^{x_1}, y_0 = g^{x_0}$, his identity and other attributes.*

C2 *The $s+1$ tuple:*

$$(A, B_{s-1}, B_{s-2}, \dots, B_0) = (\gamma^k, x_{s-1}^{-1} \cdot [\psi^k]_{s-1}, x_{s-2}^{-1} \cdot [\psi^k]_{s-2}, \dots, x_0^{-1} \cdot [\psi^k]_0),$$

where $0 < k < \omega$ is randomly chosen by the user.

C3 *A transcript proving that the $x_{s-1}, x_{s-2}, \dots, x_1, x_0$ appearing in component C1, are also appearing in component C2, i.e. are recoverable from it by the KRA.*

Observe that, provided that the components **C1** and **C2** are correctly formed, then the KRA can first recover ψ^k from A , then x_i^{-1} and thus x_i from B_i ($i = s-1, \dots, 0$). The transcript in C3, consists of two parts. The first part is the $(t-s)$ -tuple

$$(d_{t-1}, d_{t-2}, \dots, d_s) = (g^{[\psi^k]_{t-1}}, g^{[\psi^k]_{t-2}}, \dots, (g^{[\psi^k]_s})). \quad (5)$$

The second part is a transcript of a non-interactive proof that there exists a $0 < k < \omega$ such the following two equalities in power polynomials hold:

$$\begin{aligned} g^{(\psi^k)} &= d_{t-1}X^{t-1} + d_{t-2}X^{t-2} + \dots + d_sX^s + \\ &\quad + y_{s-1}^{B_{s-1}}X^{s-1} + y_{s-2}^{B_{s-2}}X^{s-2} + \dots + y_0^{B_0} \\ \gamma^k &= A \end{aligned} \quad (6)$$

If the above two equalities hold for some k , then this shows that **C1** and **C2** are correctly formed, i.e. contain the same x_i for $i = s - 1, \dots, 0$. It is clear from Section 3 how to form the transcript in the schemes as mentioned in Scheme 41. With respect to the choice of the security parameter v , 2^{-60} seems like a reasonable probability of failure. Moreover, certification is usually an on-line process, in which a user authenticates himself using an access code given by an RA. Hence one can easily incorporate an online challenge of the Certification Authority in the input of the hash resulting in the challenges h_1, \dots, h_v , and puts a maximum time on the certification session, then 2^{40} seems like a reasonable bound for K . This means that $v = 100$ seems like a reasonable choice for the security parameter.

As an illustration, suppose that G is a group consisting of a collection of points on an elliptic curve of prime order p of 160 bit length, then one can compare this with the security of the ElGamal encryption system in the multiplicative group of a basic field of size 1600 bits, cf. [14]. So to get the same level for security for the KRA - as in the construction of CRs in [27] - one could employ our subgroup variant in $GF(p^t)$ with $t = 10$. This would result in a CR of about 64,000 bits, much smaller than the length of a comparable CR in the schemes in [27], which would be of size 160,000 bits. Observe that $t = 10$ also facilitates the use of optimal normal bases (as mentioned in Section 2.2), making our scheme also more efficient than comparable schemes in [27]. To get more security for the KRAs, say “1900 bits”, one could use a similar technique in $GF(p^t)$ with $t = 12$. This would then still result in a CR of 64,000 bits, still smaller than the length of the above mentioned transcript.

4.2 Security Analysis of our Scheme

With respect to security, let us first look at Scheme 41 with $s = t$. Then the first part of component **C3** (mentioned in formula (5)) is empty, and it follows that **C3** is a transcript of a proof of knowledge from which no secret information can be extracted. Now, an attacker could try to find one of the user’s private keys, x_i say, using one of the following three strategies. First, he could try to find x_i from component **C1**, which would mean breaking the discrete logarithm in G . Secondly, he could try to find x_i from component **C2**, which would mean breaking the Diffie-Hellman problem in Γ by Corollary 22. This problem is designed to be more difficult (in a scalable fashion) than to break the public key of the user.

Finally, he could try to find x_i from combining the information from the components **C1** and **C2**. As the discrete logarithm problems employed in **C1** and **C2** are only related by the number p , it seems unlikely that combining both components **C1** and **C2** will be very beneficiary. We’ll now briefly discuss a formal result in this direction.

We recall from Section 2.1 that a suitable generator γ of the multiplicative group of $GF(p^t)$ should have a prime subgenerator that is not contained in one of the proper subfields and should have a suitably large prime order ϖ .

In our constructions, it is very likely that the power of ϖ in the primenumber decomposition of $p^t - 1$ is one, i.e. $p^t - 1/\varpi$ and ϖ are relatively prime. One can uniquely decompose γ as the product $\gamma' \cdot \gamma''$ where the order of γ' equals ϖ and that of γ'' equals $p^t - 1/\varpi$.

The following result is a generalization of [24, Proposition 1].

Theorem 42 *We use the above notation. Let the ElGamal encryptions used for the KRA be classical, i.e. the generator γ generates the whole multiplicative group of $GF(p^t)$ and let i be one of $0, \dots, t - 1$. Also, let the power of ϖ in the primenumber decomposition of $p^t - 1$ be one.*

*Then, using an efficient algorithm \mathcal{P} computing x_i from **C1** and **C2**, one can either construct*

1. *an efficient algorithm \mathcal{P}_{DL} solving the discrete logarithm problem in G , or*
2. *an efficient algorithm \mathcal{P}_{DDH} solving the Decision-Diffie-Hellman problem w.r.t. the base γ' .*

Sketch of Proof: Suppose that \mathcal{P} is an efficient algorithm computing x_i from **C1** and **C2**. Consider a triple (A, B, C) in $\langle \gamma' \rangle^3$. Using the technique appearing in the proof of [24, Proposition 1], one can transform this to a random triple $(\hat{A}, \hat{B}, \hat{C})$ in $\langle \gamma' \rangle^3$ that is either a random Diffie-Hellman triple if (A, B, C) is, or a random non-Diffie-Hellman triple otherwise.

Moreover, by suitably multiplying $\hat{A}, \hat{B}, \hat{C}$ with random Diffie-Hellman triples in $\langle \gamma'' \rangle^3$, one obtains a triple $(\bar{A}, \bar{B}, \bar{C})$ in the multiplicative group of $GF(p^t)$, that is either a random Diffie-Hellman triple if (A, B, C) is, or a random non-Diffie-Hellman triple, with respect to the element γ' otherwise. Hence, $(\bar{A}, \bar{B}, \bar{C})$ is always a random Diffie-Hellman triple with respect to γ'' .

Using the technique appearing in the proof of [24, Proposition 1], one can now either construct

1. *an efficient algorithm \mathcal{P}_{DL} solving the discrete logarithm problem in G , or*
2. *an efficient algorithm $\mathcal{P}_{DDH(\gamma')}$ solving the Decision-Diffie-Hellman problem w.r.t. the base γ' .*

□

Let us assume that the DH decision problem in $\langle \gamma' \rangle$ is as hard as the DL problem in $\langle \gamma' \rangle$, which is constructed to be more difficult than the DL problem in G . Then we can conclude from the previous result that the best strategy an attacker can follow to determine x from **C1** and **C2** is to break the DL problem in G . Hence, at least from a *theoretical* point of view, using the classical ElGamal scheme in our scheme is more secure than using the subgroup variant. However, we do not expect that using the subgroup variant of the ElGamal in our scheme is less secure in practice than using the classical ElGamal scheme.

We are left with a general security proof of the Scheme 41, let $s = s_0$ with $1 \leq s_0 < t$. Now, the idea is that we will construct an arbitrary “output” of our scheme with $s = s_0$ from an arbitrary “output” with $s = t$. So, if one has an

advantage in breaking the scheme with $s = s_0$, one also has this advantage for breaking the scheme with $s = t$. As we have shown the security for the scheme with $s = t$ this would conclude the security proof of the scheme with $s = s_0$ also.

For this construction, let **C1**, **C2** and **C3** be an arbitrary output for the scheme with $s = t$. Forming **C1'** and **C2'** for the scheme with $s = s_0$ simply means removing some information from **C1** and **C2**. Also, from the transcript in **C3'** one obtains a representation of the power polynomial $g^{(\psi^k)}$ (see equality (6)). So we can form the first part $(d_{t-1}, d_{t-2}, \dots, d_s)$ of **C3'** by simply taking the highest $t - s$ coefficients of the power polynomial $g^{(\psi^k)}$. Finally, the second part of **C3'** must be a random transcript that proves that equality (6) holds. However, the second part of **C3** provides us with such a random transcript.

4.3 Improvements and Generalizations of our Scheme

Reduced Ciphertext In our basic scheme, the user has total freedom in choosing his private keys which might be useful in some situations (e.g. when using already existing public keys). However, if there are no restrictions on the user's private keys other than that they should be uniformly random, then the component **C2** can be reduced, like is done in [27]. Let us assume that the number of user public keys equals the extension degree of the field, i.e. $s = t$; the case $s < t$ is a straightforward generalization.

In this situation, the user first generates γ^k where $0 < k < \omega$ is randomly chosen and then chooses the private x_i as $[\psi^k]_i$, and forms his public keys as $y_i = g^{x_i}$ as usual ($i = 0, \dots, s - 1$). The user then forms components **C1** and **C2**, the latter of which only needs to contain γ^k as $x_i^{-1} \cdot [\psi^k]_i = 1$. The component **C3** is formed as before. Note the following equality for power polynomials holds (modulo the fixed irreducible polynomial F):

$$g^{(\psi^k)} = y_{t-1}X^{s-1} + y_{s-2}X^{s-2} + \dots + y_0.$$

That is, the t public keys of the user can be represented by the power polynomial $g^{(\psi^k)}$. Hence, the components **C1**, **C2** in the improved scheme, simply take the form

$$g^{(\psi^k)}, \gamma^k.$$

And **C3** is a non-interactive proof showing that k occurs in both components. It easily follows that this improved scheme is a secure as the basic one. and that the output of the improved scheme is $s \cdot \omega$ bits smaller than the basic scheme.

Reduced Risk of Shadow Key Establishment It is not hard to leak 20 bits of information in each of the y_i appearing in the component **C1** in the basic scheme as they can be independently chosen. So, if $t = 12$ then this would mean that 240 bits subliminal bits can be leaked in the component **C1**, which opens the possibility of the establishment of shadow public keys. Of course, this problem arises with any private key escrow scheme where the user can construct

his public key himself and which allows for several, say 12, (certified) public keys to be used. Nonetheless, using the above described improved scheme reduces this risk, as the user public keys are no longer generated independently.

Incorporating Secret Sharing In all our schemes so far, we have used a public key ψ of which the private part x_K was in the possession of only one Key Recovery Agent. It is well-known that Scheme 41 can very conveniently support the use of a KRA's public key in which the private key is secretly shared among n share-holders, i.e. sub-KRAs. For instance, suppose that all sub-KRAs have chosen a private key $0 < \theta_i < \omega$, and a public key $\psi_i = \gamma^{\theta_i}$. Then the product ψ of the latter will be the shared KRA public key; it easily follows that private keys of users can be reconstructed without the sub-KRAs having to come together. Indeed, the i -th sub-KRA forms (using the terminology from Scheme 41) $A_i = A^{\theta_i}$ and sends this to the entity requesting recovery of the user's private keys. Of course, the privilege and execution of this should be in accordance with the key recovery policy. This entity forms the product of these A_i which equals ψ^k from which the user's private keys can be retrieved.

In the subgroup variant of our schemes where ω is a prime number, further convenient improvements are possible. Here participating sub-KRA's can, without needing a trusted dealer, for any $1 \leq k \leq n$, construct a public key ψ in such a way that if it is used in our basic scheme, then the user's private key can be reconstructed only if k out of n KRA sub-holders cooperate. See [20] and [21]. Note that if one wants to use this technique to implement a k out of n key recovery scheme, then it is required that the sub-KRAs generate $\binom{n}{k}$ public keys, which grows exponentially in k .

An improvement by Schoenmakers in [22] solves a similar problem in the original CR scheme of Young and Yung [27]. However, this improvement still has the drawbacks mentioned in the introduction, specifically the lack of scalable security. By combining the techniques in [22] with ours, we can resolve these drawbacks.

More specifically, this combination offers:

- scalable security,
- the possibility that users themselves can choose a number, n , of KRAs from a given list consisting of l KRAs ($n \leq l$), and can choose a number $k \leq n$ such that k out of n of the chosen KRAs are required to retrieve the users private keys,
- the ability for users to form Certificates of Recoverability proving the correctness of the previous point to the CA.

Of course, the choice of k and n the user makes should be in accordance with the Certificate Policy under which the CA operates: if this choice is not acceptable, then the CA will not issue an certificate.

To this end, without loss of generality we again assume that $s = t$. Using the technique of [22] in our schemes, means that the private part, i.e. $\log_{\gamma}(\psi)$ of the

public key ψ is “virtual”, i.e. it is not known by anybody, and that all involved parties are certain of that. Or, in other words, γ and ψ are independently selected generators. Moreover, each i -th sub-KRA has a public key ξ_i , with respect to ψ (and not to γ as before), i.e. $\xi_i = \psi^{\theta_i}$ where θ_i is the private key of the sub-KRA. Now, the user encrypts his private keys using the improved scheme described above using the virtual public key ψ . As mentioned above the components \mathbf{C}_1 , \mathbf{C}_2 then take the form

$$g^{(\psi^k)}, \gamma^k$$

And \mathbf{C}_3 is a non-interactive proof showing that k occurs in both components. Of course, as ψ is a virtual public key, nobody is able to retrieve the user’s private keys from these components. This is why the user supplements these components by using the non-interactive Public Verifiable Secret Sharing scheme in [22], secretly sharing ψ^k among the KRA’s, using the public keys ξ_i where γ^k is used as the initial commitment (C_0 in the terminology of [22]). We refer to [22] for further technical details. One can easily prove that this supplementation to our schemes does not weaken security.

5 Conclusion

We have proposed two schemes for Certificates of Recoverability, making it possible for a PKI user to escrow its private keys in a publicly verifiable way, by means of encrypting it with a Key Recovery Agent’s public key and depositing this with any other party. In our schemes, the cryptographic security employed by the Key Recovery Agents can be set higher, in a scalable fashion, than that being employed by the users. Among the other improvements of our scheme are its applicability to create CRs for cryptosystems based on the Discrete Log problem in small subgroups such as Elliptic Curve Cryptosystems. Also, the size of the constructed proofs of knowledge can be taken smaller. We have additionally shown several ways to support secret sharing in our scheme. Finally, we have also presented several new constructions and results on the hardness of “small parts”, in the setting of Diffie-Hellman keys in extension fields.

Acknowledgements

Berry Schoenmakers and an anonymous referee are gratefully acknowledged for their constructive comments.

References

1. M. Adleman, J. DeMarrais *A subexponential algorithm over all finite fields*, CRYPTO ’93 Proc., Springer-Verlag, 147-158. [262](#)
2. N. Asokan, V. Shoup, M. Waidner, *Optimistic Fair Exchange of Digital Signatures*, Eurocrypt’98 Proc., Springer-Verlag, 591-606. [260](#)
3. M. Bellare, P. Rogaway, *Random Oracles are Practical: A paradigm for Designing Efficient Protocols*, 1st ACM Conference on Computer and Communications Security, ACM Press, 1993, 62-73. [268](#)

4. D. Boneh, R. Venkatesan, *Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes*, CRYPTO'96 Proc. Springer-Verlag, 129-142. [263](#)
5. A.E. Brouwer, R. Pellikaan, E.R. Verheul, *Doing More with Fewer Bits*, Asiacrypt'99 Proc., Springer-Verlag. [266](#)
6. D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Trans. on IT, 30, 1984, 587-594. [262](#)
7. W. Diffie, M.E. Hellman, *New directions in cryptography*, IEEE Trans. on IT 22, 1976, 644-654. [261](#)
8. T. ElGamal, *A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms*, IEEE Trans. on IT 31(4), 1985, 469-472. [259](#), [261](#)
9. A. Fiat, A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, CRYPTO'86 Proc., Springer-Verlag, 186-194. [268](#)
10. D.M. Gordon, *Discrete Logarithms in GF(p) using the number field sieve*, SIAM J. of Discrete Math., 6, 124-138. [260](#)
11. J. Håstad, *On Using RSA with Low Exponent in a Public Key Network*, CRYPTO'85 Proc., Springer-Verlag, 403-405. [259](#)
12. J. Kilian, F.T. Leighton, *Fair Cryptosystems Revisited*, Crypto'95 Proc., Springer-Verlag, 208-221. [261](#)
13. A.K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields*, ACISP97 Proc., Springer-Verlag, 127-138. [262](#), [263](#), [265](#)
14. A.K. Lenstra, E.R. Verheul *Selecting Cryptographic Key Sizes*, these proceedings. [270](#)
15. H.W. Lenstra, *Finding isomorphisms between two finite fields* Math. of Comp., 56 (1991), 329-347. [266](#)
16. R. Lidl, H. Niederreiter, *Finite Fields*, Addison-Wesley, 1983. [264](#)
17. S.C. Pohlig, M.E. Hellman, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Trans. on IT, 24 (1978), 106-110. [262](#)
18. J.M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. of Comp., 32 (1978), 918-924. [262](#)
19. M. Naor, M. Yung, *Universal one-way functions and their cryptographic applications*, In 21st Annual ACM Symposium on Theory of Computer Science, 1997. [266](#)
20. T.P. Pedersen, *Distributed Provers with Applications to Undeniable Signatures*, Eurocrypt'91 Proc., Springer-Verlag, 221-242. [273](#)
21. T.P. Pedersen, *A Threshold Cryptosystem Without a Trusted Party*, Eurocrypt '91 Proc., Springer-Verlag, 522-526. [273](#)
22. B. Schoenmakers, *A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting*, CRYPTO'99 Proc., 148-164. [273](#), [274](#)
23. C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, 4, 1991, 161-174. [260](#)
24. M. Stadler, *Publicly Verifiable Secret Sharing*, Eurocrypt'96 Proc., 190-199. [259](#), [261](#), [267](#), [271](#)
25. M. Stadler, J.-M. Piveteau, J. Camenisch, *Fair Blind Signatures*, Eurocrypt'95 Proc., 209-219. [261](#)
26. D.R. Stinson *Cryptography: theory and practice*, CRC press, 1995. [259](#), [260](#)
27. A. Young, M. Yung, *Auto-Recoverable Auto-Certifiable Cryptosystems*, Eurocrypt'98 Proc., 16-31. [259](#), [260](#), [261](#), [268](#), [270](#), [272](#), [273](#)

Design Validations for Discrete Logarithm Based Signature Schemes

Ernest Brickell¹, David Pointcheval², Serge Vaudenay³, and Moti Yung⁴

¹ Intel Inc., Portland, OR, USA

² CNRS-LIENS, Paris, France

³ EPFL, Lausanne, Switzerland

⁴ Certco, New York, NY, USA

Abstract. A number of signature schemes and standards have been recently designed, based on the Discrete Logarithm problem. In this paper we conduct design validation of such schemes while trying to minimize the use of ideal hash functions. We consider several Discrete Logarithm (DSA-like) signatures abstracted as generic schemes. We show that the following holds: “if the schemes can be broken by an existential forgery using an adaptively chosen-message attack then either the discrete logarithm problem can be solved, or some hash function can be distinguished from an ideal one, or multi-collisions can be found.” Thus, for these signature schemes, either they are equivalent to the discrete logarithm problem or there is an attack that takes advantage of properties which are not desired (or expected) in strong practical hash functions (SHA-1 or whichever high quality cryptographic hash function is used). What is interesting is that the schemes we discuss include KCDSA and slight variations of DSA.

Further, since our schemes coincide with (or are extremely close to) their standard counterparts they benefit from their desired properties: efficiency of computation/space, employment of certain mathematical operations and wide applicability to various algebraic structures. We feel that adding variants with strong validation of security is important to this family of signature schemes since, as we have experienced in the recent past, lack of such validation has led to attacks on standard schemes, years after their introduction. In addition, schemes with formal validation which is made public, may ease global standardization since they neutralize much of the suspicions regarding potential knowledge gaps and unfair advantages gained by the scheme designer’s country (e.g. the NSA being the designers of DSA).

1 Introduction

One of the greatest achievements of Public-key Cryptography, introduced by Diffie and Hellman [9], is the provision of a strong (non-repudiated) integrity function known as “digital signature.” The research regarding digital signature schemes has taken a number of basic directions. The first one was theoretical and engaged in reducing the computational assumption required for signature, in order to understand the inherent nature of the primitive. Indeed, digital signature

turned out to be equivalent to one-way functions [19,27]. Another direction has produced various flavors of signatures (blind, undeniable, fail-stop, etc.). The third direction was the design and standardization of efficient signature schemes which are very practical.

As part of this third direction, one avenue of research and technology development is the design of digital signature schemes based on the hardness of the discrete logarithm problem (which started with the introduction of the El Gamal signature scheme [10]). A number of efficient schemes have appeared since then and a few of them were standardized, in particular NIST standardized the DSA signature scheme (DSS) [20].

Whereas the theoretical signature schemes have been presented with a security proof against (existential forgery) attacks [14,15], the practical schemes were given in an ad-hoc fashion based on intuitive feeling of security. However, as we know in the past and the recent future many schemes believed to be secure have been later broken. Thus the situation is not satisfactory w.r.t. the practical schemes.

When attacking the security of the practical scheme one may attempt a security proof from scratch based solely on computational assumptions. This may not be easily doable since the typical scheme is very specific and it typically employs hashing in addition to involving various operations which were not guided by any structure.

The next proof direction is to assume that certain hash functions which are available and which everyone has a black-box access to, are ideal (i.e., are like a random oracle [2]). Since the available hash functions cannot in fact be random oracles but rather computationally indistinguishable from one, the proof becomes an argument for security: As long as there is no evidence that distinguishes the hash function in use from a random oracle, the security of the scheme is reduced to a well defined number theoretic problem (namely the discrete logarithm). Of course, one has to be careful here. First, only hash functions which are used as a black-box and are replaceable (in case of a specific weakness is found) should be assumed to “look random.” Secondly, the methodology does not work universally for every scheme: an artificial theoretical construction was shown which is provably secure under random oracle assumption but becomes insecure under any “concrete implementation” of the oracle [8]. Luckily, this is only an example, and its structure does not apply to the practical signature schemes we study (intuitively, in these schemes there is a separation of the role of the hash function from that of the number theoretic function, such as discrete logarithm).

It is believed that the “random oracle” proof methodology still gives a much better understanding and confidence in a scheme than if a scheme is left completely unanalyzed. In the latter case, any unexpected attack may be mounted, whereas the security study assuming ideal hash, greatly limits the potential attack scenarios.

In this work we study, under the random oracle model, but minimizing the random oracle use, schemes which are, or very closely related to, the standardized schemes (DSA [20], KCDSA [17]). Our goal is to exploit the efficiency of these

schemes, yet to modify them slightly if necessary in order to claim validation of security. We believe that the modified schemes are within the spirit of the standards, yet have a strength of being validated. We believe that perhaps the standard bodies should look carefully into our study.

The random oracle methodology was first employed informally by Fiat and Shamir [12], and formalized in Bellare and Rogaway [2]. It was used in showing variants of RSA signatures [26,4]. At the same time Pointcheval and Stern [23] formalized the Fiat-Shamir technique and then validated security for an El Gamal variant signature, Schnorr signature [28,29] and Fiat-Shamir signatures. In so doing they formalized the “forking lemma” methodology which we will follow. Further analysis and investigation of multi-signature was performed by Ohta and Okamoto [22]. However, the case of variants of the standardized DSA-like signature which we analyze herein was left open.

Outline of the Paper. In the next Section we present our basic definitions and in Section 3, the basic signature schemes and variants we deal with, namely El Gamal-type signature schemes. In section 4 we present the generic schemes we prove security about, together with some concrete examples. Section 5 presents the basic security result, and its proof. Example of how to apply the general results to the variants of the standard schemes is given in Section 6, where Section 7 concludes the work.

Remark. The paper is based on a merge, crystallization and generalization of initial ideas reported in our earlier studies [7,24]. The current version contains improved and more elegant analysis as well as the important direction of minimizing the use of the “random oracle” assumption.

2 Definitions

In this section, we recall security notions for signature schemes and hash functions and review the *random oracle model*.

2.1 Security Notions for a Signature Scheme

We first define a signature scheme and then the notions of security. Similar definitions can be found in [15,23], where the reader is referred for more details.

Definition 1 (A Signature Scheme). A signature scheme *consists of three polynomial time randomized algorithms, (Key-Gen, Sign, Ver)*.

- *Key-Gen* takes as input a random string and outputs a pair of keys (X, Y) , where X is the private signature key, and Y is the public verification key.
- *Sign* takes as input a message M and the private signature key X , and produces a signature Sig .
- *Ver* takes as input a message M , a signature Sig , the public verification key Y , and checks whether Sig is a valid signature of M .

We will use the definition of security of a signature scheme defined by [15], known as existential unforgeability against adaptively chosen-message attacks.

Definition 2 (Unforgeability). *We say that a signature scheme (Key-Gen , Sign , Ver) is unforgeable if no adversary who is given the public verification key Y , and the signatures of k messages, M_1, \dots, M_k adaptively chosen by herself, can produce the signature on a new message M with non-negligible probability.*

2.2 Security Notions for a Hash Function

A hash function is any function which takes as input a message of any length and outputs a digest of fixed size (typically 128 or 160 bits).

Definition 3 (Multi-Collision-Freeness). *A function h is said ℓ -collision-free, if there is no ℓ -tuple (x_1, \dots, x_ℓ) of pairwise distinct elements such that $h(x_1) = \dots = h(x_\ell)$.*

But for a hash function, which takes variable (any)-length inputs, absence of multi-collisions can not be guaranteed, but perhaps we can hope for the impossibility of finding some of them.

Definition 4 (Multi-Collision-Resistance). *A function h is said ℓ -collision-resistant, if it is computationally impossible to find an ℓ -tuple (x_1, \dots, x_ℓ) of pairwise distinct elements such that $h(x_1) = \dots = h(x_\ell)$.*

For simplicity, a *collision-resistant* hash function is, in general, a 2-collision-resistant hash function.

2.3 The Random Oracle Model

In many signature schemes, a cryptographic hash function, such as MD5 [25] or SHA-1 [21], is used, namely to reduce the size of the message. Such a cryptographic hash function has the property that it is collision-resistant, and therefore one-way.

Many recent proofs [3,4,22,23] make the assumption that this cryptographic hash function is an *ideal random function* also known as *random oracle*: for any new query, the answer is uniformly distributed in the output set, independently of previous query/answer pairs. This is the so-called *random oracle model* [2].

Moreover, in this model, a simulator is allowed to set the output of the random oracle to specific values (uniformly distributed) for an input that had not yet been defined. Such a property will be required in the following.

However, since proofs in the random oracle model are just security arguments, but not the strongest proof of security that one could require, we try to minimize the use of random oracles.

3 The El Gamal Type Signature Schemes

El Gamal [10] was the first to propose a signature scheme based on the discrete logarithm problem. Then, Schnorr [28,29] improved the scheme using the modulo q truncating function, playing in a prime subgroup. This fixes some weaknesses later on discovered by Bleichenbacher [5], van Oorschot and Wiener [30] and also discussed by Anderson and Vaudenay [1]. This latter scheme has been formally proven unforgeable in the random oracle model relative to the discrete logarithm problem [22,23]. However, as discussed in the introduction many other variants have been defined and standardized by governments: the US-standard DSA [20] and the Korean-standard KCDSA [17]. In both cases, there are system parameters p, q, g such that q and p are primes, $q|p - 1$, and g is an element of order q in the group \mathbb{Z}_p^* , i.e. the group of invertible integers modulo p . A user has a public key Y , and a private key X such that $Y = g^X \bmod p$.

3.1 The DSA Signature

The Digital Signature Algorithm [20] has been standardized by the US government, together with the hash function SHA-1 [21], denoted by H in the following description. To sign a message M , the **Sign** algorithm picks a random invertible element k in \mathbb{Z}_q^* and computes

$$\begin{aligned} R &= g^k \bmod p & T &= R \bmod q \\ U &= H(M) & S &= (U + XT)/k \bmod q \end{aligned}$$

Formally, the random tape ω defines k and $\text{Sign}(\omega, M) = (S, T)$. The **Ver** algorithm consists of checking whether

$$\left(g^{\frac{U}{S}} Y^{\frac{T}{S}} \bmod p \right) \bmod q = T \text{ or not, where } U = H(M).$$

3.2 The KCDSA Signature

The Korean Certificate-based Digital Signature Algorithm [17] has recently been standardized by the Korean government, and is proposed to the IEEE P1363a, as a signature standard. It uses two hash functions G and H . We note that, for efficiency concern, $1/X \bmod q$ is considered as private key instead of X . To sign a message M , the **Sign** algorithm picks a random element k in \mathbb{Z}_q^* and computes

$$\begin{aligned} R &= g^k \bmod p & T &= G(M) \\ U &= H(R) & S &= (k - T \oplus U)/X \bmod q. \end{aligned}$$

Formally, the random tape ω defines k and $\text{Sign}(\omega, M) = (S, U)$. The **Ver** algorithm consists of first checking the sizes of S and U . Then computing

$$E_G = T \oplus U \quad W = g^{E_G} Y^S \bmod p$$

and checking whether or not $U = H(W)$, where $T = G(M)$.

3.3 DSA Variants

In order to provide better analyzable schemes than DSA, one can study the following variants, we call DSA–I and DSA–II respectively, which only slightly differ from the original one.

The DSA–I Variant. This first variant differs from the original scheme just by replacing the “ $x \mapsto x \bmod q$ ” truncating function by any hash function G . To sign a message M , the **Sign** algorithm picks a random invertible element k in \mathbb{Z}_q^* and computes

$$\begin{aligned} R &= g^k \bmod p & T &= G(R) \\ U &= H(M) & S &= (U + XT)/k \bmod q. \end{aligned}$$

Formally, the random tape ω defines k and $\text{Sign}(\omega, M) = (S, T)$. The **Ver** algorithm consists of checking whether

$$G\left(g^{\frac{U}{S}} Y^{\frac{T}{S}} \bmod p\right) = T \text{ or not, where } U = H(M).$$

The second variant is a bit more different from DSA, but it requires weaker (more acceptable) assumptions whenever possible.

The DSA–II Variant. To sign a message M , the **Sign** algorithm picks a random invertible element k in \mathbb{Z}_q^* and computes

$$\begin{aligned} R &= g^k \bmod p & T &= G(R) \\ U &= H(M, T) & S &= (U + XT)/k \bmod q. \end{aligned}$$

Formally, the random tape ω defines k and $\text{Sign}(\omega, M) = (S, T)$. The **Ver** algorithm consists of checking whether

$$G\left(g^{\frac{U}{S}} Y^{\frac{T}{S}} \bmod p\right) = T \text{ or not, where } U = H(M, T).$$

We note that, based on [24], this second variant has already been included in the ISO/IEC 14888 report [16]. We further note that below we will also give arguments for security when G is the $\bmod q$ function.

3.4 Security

One can prove relatively easily that the DSA–I variant is unforgeable [7,24] relative to the discrete logarithm problem assuming that both G and H are random oracles. However, this is a very strong assumption which has no real practical impact to the original DSA. Indeed, while the (easily replaceable) SHA-1 function can be assumed practically “ideal”, as it is usually done in the random oracle based papers [2,3,4,22,23], the “ $x \mapsto x \bmod q$ ” map of DSA cannot be assumed

random due to its algebraic properties. Similarly, KCDSA can be investigated in the full “random oracle” model [24].

In the following, we formally define two (general) families of El Gamal-type signature schemes which include the above variants (KCDSA and DSA-II), but may be used to guide future designs as well. We then provide security proofs assuming some generic hash functions to be ideal random ones while assuming that some others are just (multi)-collision-resistant/free. This will provide validation related to the DSA and KCDSA national standards.

4 The Trusted El Gamal Type Signature Schemes

For the two types of schemes defined in the following, there are

- system parameters p, q, g such that q and p are primes, $q|p - 1$, and g is an element of order q in the group \mathbb{Z}_p^* , i.e. the group of invertible integers modulo p .
- two hash functions G and H , whose output ranges are denoted by \mathcal{G} and \mathcal{H} respectively. We assume that $q/2 < |\mathcal{G}|, |\mathcal{H}| < 2q$. In both cases, H is considered as an *ideal random function* (or a random oracle), whereas G only requires some practical properties, such as (multi)-collision-resistance or (multi)-collision-freeness.
- three functions:

$$F_1 : (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \longrightarrow \mathbb{Z}_q \quad F_2 : (\mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \longrightarrow \mathbb{Z}_q \quad F_3 : (\mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \longrightarrow \mathbb{Z}_q$$

satisfying for all $(a, b, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{G}, \mathcal{H})$,

$$F_2(F_1(a, b, T, U), T, U) + b \cdot F_3(F_1(a, b, T, U), T, U) = a \bmod q.$$

In addition, each user has private and public keys X, Y , such that $Y = g^X \bmod p$.

Definition 5 (The TEGTSS Verification Equation). A tuple (W, S, T, U) is said to satisfy the TEGTSS verification equation if for $E_G = F_2(S, T, U)$, and $E_Y = F_3(S, T, U)$ then $W = g^{E_G} Y^{E_Y} \bmod p$.

Then, Trusted El Gamal Types Signature Schemes are of two distinct types, depending on the use of the functions G and H .

4.1 Type I: the TEGTSS-I Schemes.

- To sign a message M , the signer chooses an element k at random in \mathbb{Z}_q^* , computes $T = G(M)$ and generates $R = g^k \bmod p$. He then gets $U = H(R)$ and computes $S = F_1(k, X, T, U)$.
The signature of M is the triple (S, T, U) . In practice, the pair (S, U) is enough since $T = G(M)$, but we keep the triple for the reader’s convenience.
- To verify the signature (S, T, U) of the message M , a verifier computes $E_G = F_2(S, T, U)$ and $E_Y = F_3(S, T, U)$ and finally $W = g^{E_G} Y^{E_Y} \bmod p$. He then checks whether $T = G(M)$ and $U = H(W)$ or not.

TEGTSS–I Properties. To provide a TEGTSS–I scheme, F_3 must satisfy the following conditions for tuples (W, S_i, T_i, U_i) for $i = 1, 2$ that satisfy the *TEGTSS Verification Equation*:

1. if $T_1 \neq T_2$, then $F_3(S_1, T_1, U_1) \neq F_3(S_2, T_2, U_2)$.
2. For a fixed verifying tuple (W, S_1, T_1, U_1) there is a one-to-one map between the values of U_2 and the values of T_2 such that (W, S_2, T_2, U_2) verifies the TEGTSS equation and $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$.

Example: the KCDSA Signature. Both signature and verification algorithms are exactly as described above, with the following functions:

$$F_1(k, X, T, U) = (k - T \oplus U)/X \bmod q$$

$$F_2(S, T, U) = T \oplus U \bmod q \text{ and } F_3(S, T, U) = S \bmod q,$$

where $T = G(M)$, $R = g^k \bmod p$, $U = H(R)$ and $S = F_1(k, X, T, U)$.

Lemma 6. *The KCDSA signature is a TEGTSS–I scheme.*

Proof. We need to show that the functions F_1 , F_2 and F_3 satisfy the properties of a TEGTSS–I scheme:

- for all $(k, X, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{Q}, \mathcal{H})$,
- $$\begin{aligned} F_2(F_1(k, X, T, U), T, U) + X \times F_3(F_1(k, X, T, U), T, U) \\ = T \oplus U + X \times (k - T \oplus U)/X = k \bmod q, \end{aligned}$$
- if $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$, then $S_1 = S_2 = S$ and since W is fixed, $F_2(S, T_1, U_1) = F_2(S, T_2, U_2)$ and $U_1 = U_2$. Therefore $T_1 \oplus U_1 = T_2 \oplus U_2$ and consequently $T_1 = T_2$.
 - suppose $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$ for a given W . Then $S_1 = S_2 = S$. Since W is fixed, $F_2(S, T_1, U_1) = F_2(S, T_2, U_2)$: $T_1 \oplus U_1 = T_2 \oplus U_2$. Therefore $T_2 = T_1 \oplus U_1 \oplus U_2$.

□

4.2 Type II: the TEGTSS–II Schemes.

- To sign a message M , the signer chooses an element k at random in \mathbb{Z}_q^\star , computes $R = g^k \bmod p$ and $T = G(R)$. He then gets $U = H(M, T)$ and computes $S = F_1(k, X, T, U)$.
The signature of M is the triple (S, T, U) . In practice, the pair (S, T) is enough, since $U = H(M, T)$. But the triple is kept for the reader's convenience.
- To verify the signature (S, T, U) on the message M , a verifier computes $E_G = F_2(S, T, U)$ and $E_Y = F_3(S, T, U)$ and finally $W = g^{E_G} Y^{E_Y} \bmod p$. He then checks whether $T = G(W)$ and $U = H(M, T)$ or not.

TEGTSS–II Properties. To provide a TEGTSS–II scheme, the functions F_2 and F_3 must satisfy the following one-to-one condition: for given T , E_G and E_Y , there exists a unique pair (U, S) such that

$$E_G = F_2(S, T, U) \text{ and } E_Y = F_3(S, T, U).$$

Furthermore, this pair is easy to find.

Example: the DSA-II Variant. This DSA variant is exactly as described above, with the following functions:

$$F_1(k, X, T, U) = (U + XT)/k \bmod q$$

$$F_2(S, T, U) = U/S \bmod q \text{ and } F_3(S, T, U) = T/S \bmod q,$$

where $R = g^k \bmod p$, $T = G(R)$, $U = H(M, T)$ and $S = F_1(k, X, T, U)$.

Lemma 7. *The DSA-II signature is a TEGTSS-II scheme.*

Proof. We need to show that the functions F_1 , F_2 and F_3 satisfy TEGTSS-II properties:

- for all $(k, X, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{Q}, \mathcal{H})$,
- $$\begin{aligned} F_2(F_1(k, X, T, U), T, U) + X \times F_3(F_1(k, X, T, U), T, U) \\ = U/S + XT/S = (U + XT)/S = k \bmod q, \end{aligned}$$
- for given T , E_G and E_Y , $F_2(S, T, U) = E_G$ and $F_3(S, T, U) = E_Y$ imply $S = T/E_Y \bmod q$ and $U = SE_G \bmod q$.

□

5 Security Results

We next claim the following security results for Trusted El Gamal Type Signature Schemes of both types.

Theorem 8. *Let us consider an attacker \mathcal{A} against a Trusted El Gamal Type Signature Scheme. Let us assume that \mathcal{A} is able to perform an existential forgery under an adaptively chosen-message attack with probability $\varepsilon > 4/q$ after Q queries to the H function.*

- for Type I schemes, if G is collision-resistant and H a random oracle, then one extracts the secret key X with less than $25Q/\varepsilon$ replays of \mathcal{A} , with constant probability greater than $1/100$.
- for Type II schemes, if G satisfies one of the following conditions,
 - G is $(\ell + 1)$ -collision-resistant
 - or, $x \mapsto G(g^x \bmod p)$ is $(\ell + 1)$ -collision-free
 and H a random oracle, then one extracts the secret key X with less than $25Q\ell \log(2\ell)/\varepsilon$ replays of \mathcal{A} (where \log denotes the logarithm is basis 2), with constant probability greater than $1/100$.

The rest of this section is devoted to the proof of the above Theorem.

5.1 General Method of Proof

We construct a simulator that produces signatures of a given message in a reasonable (poly) time in a way indistinguishable from the signer's. Therefore, if the attacker could construct a successful adaptively chosen-message attack using the legitimate signer, then she would be able to do so using only the simulator. Then we use a forking lemma as in [23] to show that if the attacker can construct a

signature with a specific ideal hash function, she can (with non-negligible probability) construct many signatures with the same fixed values, but in which the ideal hash functions output different answers. We then show that two such signatures can be used to compute the discrete logarithm of the public key, thus solving the discrete logarithm problem.

In their paper [23], Pointcheval and Stern defined particular sub-cases of the Type II signatures, where G is the identity and therefore clearly multi-collision-resistant and even collision-free. Here, we need similar tools, namely their “splitting lemma” and an improved version of their “forking lemma”. The “splitting lemma” is a formal probabilistic version of the “heavy rows lemma” [11,22].

Lemma 9 (The Splitting Lemma). *Let $A \subset X \times Y$ and we assume that $\Pr[(x, y) \in A] \geq \varepsilon$. Define*

$$B = \left\{ (x, y) \in X \times Y \mid \Pr_{y' \in Y}[(x, y') \in A] \geq \frac{\varepsilon}{2} \right\} \quad \text{and} \quad \bar{B} = (X \times Y) \setminus B,$$

then the following statements hold:

- (i) $\Pr[B] \geq \varepsilon/2$
- (ii) $\forall (x, y) \in B, \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon/2$.
- (iii) $\Pr[B \mid A] \geq 1/2$.

Proof. See Pointcheval–Stern’s [23] or Ohta–Okamoto’s [22] papers. \square

For any $\ell \leq \sqrt{q}/4$, one can state the following variant of the forking lemma [23].

Lemma 10 (The Improved Forking Lemma). *Let us consider a probabilistic polynomial time Turing machine \mathcal{A} , called the attacker, and a probabilistic polynomial time simulator \mathcal{B} . If \mathcal{A} can find with probability $\varepsilon > 4/q$ a verifiable tuple (M, R, S, T, U) with less than Q queries to the hash function, for a new message M and for a U directly defined by H , then with a constant probability $1/96$, with $(1+24Q\ell \log(2\ell))/\varepsilon$ replays of \mathcal{A} and \mathcal{B} with different random oracles, \mathcal{A} will output $\ell + 1$ verifiable tuples $(M_i, R_i, S_i, T_i, U_i)_{i=1,\dots,\ell+1}$ such that the U_i are pairwise distinct, and all the R_i equal for TEGTSS–I schemes but all the (M_i, T_i) equal for TEGTSS–II schemes.*

Proof. Let Ω and Φ denote the sets of all random tapes that could be used by the attacker \mathcal{A} and the simulator \mathcal{B} respectively, and let ω denote an arbitrary random tape in Ω , and let ϕ denote an arbitrary random tape in Φ . Let Ψ denote the set of all random tapes that define the random oracle H and let ψ denote an arbitrary random tape in Ψ . The attacker \mathcal{A} can query H directly by requesting the value of $H(X)$ for some X or \mathcal{A} can cause a query of H indirectly by asking \mathcal{B} for a signature of a message M . During the execution of the protocol, let $\mathcal{Q}_1, \dots, \mathcal{Q}_Q$, denote the ordered set of direct queries for H . We assume that $U = H(\mathcal{Q}_j)$, with $\mathcal{Q}_j = R$ for TEGTSS–I schemes, and $\mathcal{Q}_j = (M, T)$ for TEGTSS–II schemes, for some $j \leq Q$, since \mathcal{A} would have to know the random answer U to be able to determine if (M, R, S, T, U) was a verifiable tuple, excepted with probability $\nu \leq 2/q \leq \varepsilon/2$.

Therefore, the probability over the choice of ω , ϕ and ψ such that \mathcal{A} outputs a new verifiable tuple, (M, R, S, T, U) , after Q values, for a new message M and U directly defined by H (and not by the simulator \mathcal{B}) is at least $\varepsilon - \nu \geq \varepsilon/2$. We say that (ω, ϕ, j, ψ) is a winning input if for tapes (ω, ϕ, ψ) , \mathcal{A} outputs a verifiable tuple (M, R, S, T, U) after Q queries, for a new message M , in which $\mathcal{Q}_j = R$ or $\mathcal{Q}_j = (M, T)$, respectively: \mathcal{Q}_j is the crucial query.

By the Splitting-Lemma (Lemma 9), there exists a set $\Gamma \subseteq \Omega \times \Phi$ such that $\Pr[(\omega, \phi) \in \Gamma] \geq \varepsilon/4$, and for $(\omega, \phi) \in \Gamma$, $\Pr[\exists j, (\omega, \phi, j, \psi) \text{ winning}] \geq \varepsilon/4$. Furthermore,

$$\Pr[(\omega, \phi) \in \Gamma \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2}.$$

For each $(\omega, \phi) \in \Gamma$, let us define $\mathcal{J}(\omega, \phi)$ to be the set of indices $j \leq Q$ such that $\Pr[(\omega, \phi, j, \psi) \text{ winning}] \geq \varepsilon/8Q$. Since the number of possible j is upper-bounded by Q , one can easily prove by contradiction that $\mathcal{J}(\omega, \phi) \neq \emptyset$ and

$$\Pr[j \in \mathcal{J}(\omega, \phi) \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2}.$$

For $(\omega, \phi) \in \Gamma$ and $j \in \mathcal{J}(\omega, \phi)$, define a partition of the set Ψ into Hash-Classes, where a Hash-Class is defined by a tuple $(P_1, P_2, \dots, P_{j-1})$ and $\psi \in \Psi$ is in the Hash-Class $(P_1, P_2, \dots, P_{j-1})$ if $H(\mathcal{Q}_i) = P_i$ for $i \leq j-1$ for all queries \mathcal{Q}_i for $i \leq j-1$ that result from running (ω, ϕ, ψ) . Moreover, $\Psi_{\omega, \phi, j, \psi}$ will be defined to be the Hash-Class $(P_1, P_2, \dots, P_{j-1})$ where $H(\mathcal{Q}_i) = P_i$ for $i \leq j-1$ for all queries \mathcal{Q}_i for $i \leq j-1$ that result from running (ω, ϕ, ψ) .

By the Splitting-Lemma (Lemma 9), there exists a set of Hash-Classes, $\Theta(\omega, \phi, j)$ such that $\Pr[\psi \in \Theta(\omega, \phi, j)] \geq \varepsilon/16Q$ and for each $\psi \in \Theta(\omega, \phi, j)$, $\Pr[(\omega, \phi, j, \psi') \text{ winning} \mid \psi' \in \Psi_{\omega, \phi, j, \psi}] \geq \varepsilon/16Q$. Furthermore,

$$\Pr[\psi \in \Theta(\omega, \phi, j) \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2}.$$

To generate two verifiable tuples, tapes ω , ϕ are chosen at random. If \mathcal{A} does not forge a signature, then new tapes ω , ϕ are chosen, until a forgery $(M_1, R_1, S_1, T_1, U_1)$ occurs, with \mathcal{Q}_j as crucial query and Hash-Class $\Psi_{\omega, \phi, j, \psi}$. This is a “good” forgery if $(\omega, \phi) \in \Gamma$, $j \in \mathcal{J}(\omega, \phi)$ and $\psi \in \Theta(\omega, \phi, j)$, which happens with probability greater than

$$\begin{aligned} & \Pr[\psi \in \Theta(\omega, \phi, j) \mid (\omega, \phi, j, \psi) \text{ winning} \wedge (\omega, \phi) \in \Gamma \wedge j \in \mathcal{J}(\omega, \phi)] \\ & \quad \times \Pr[j \in \mathcal{J}(\omega, \phi) \mid (\omega, \phi, j, \psi) \text{ winning} \wedge (\omega, \phi) \in \Gamma] \\ & \quad \times \Pr[(\omega, \phi) \in \Gamma \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}. \end{aligned}$$

Thereafter, one fixes tapes ω , ϕ and chooses ψ' at random in Hash-Class $\Psi_{\omega, \phi, j, \psi}$. \mathcal{A} is run again for $24Q \log(2\ell)/\varepsilon$ repetitions (where \log denotes the logarithm in basis 2) or until another forgery is produced with \mathcal{Q}_j as crucial query: such an event occurs with probability greater than $(1 - 1/2\ell)$, if the first forgery was a “good” one. One repeats this process $\ell - 1$ other times, and therefore gets ℓ more forgeries with probability greater than $(1 - 1/2\ell)^\ell \geq 1/3$.

With probability greater than $(1 - \ell/|\mathcal{H}|)^\ell \geq e^{-4\ell^2/q}$, the U_i are pairwise distinct. Finally, one gets the following probability of success, if $\ell \leq \sqrt{q}/4$,

$$\begin{aligned} & \Pr \left[\begin{array}{l} \ell + 1 \text{ winning inputs after } (1 + 24Q\ell \log(2\ell))/\varepsilon \text{ trials} \\ \text{with pairwise distinct } U_i \end{array} \right] \\ & \geq \Pr[\text{a winning input after } 1/\varepsilon \text{ trials}] \\ & \quad \times \Pr[\text{a "good" winning input} \mid \text{winning input}] \\ & \quad \times \Pr \left[\begin{array}{l} \ell \text{ other winning inputs} \\ \text{after } 24Q\ell \log(2\ell)/\varepsilon \text{ trials} \end{array} \middle| \text{"good" winning input} \right] \times e^{-4\ell^2/q} \\ & \geq \frac{1}{3} \times \frac{1}{8} \times \frac{1}{3} \times \frac{3}{4} = \frac{1}{96} \geq \frac{1}{100}. \end{aligned}$$

□

In the following, we apply this forking lemma to prove the security of both families.

5.2 TEGTSS – Type I

Let us start with the Type I (which includes the KCDSA scheme), proving first the existence of an indistinguishable simulator.

Lemma 11. *Suppose H is an ideal random function with output between 0 and $|\mathcal{H}| - 1$. Then there exists a simulator that creates verifiable tuples such that after b steps, the probability that the simulator can be distinguished from a signer is less than $b^2/2q$.*

Proof. Given a message M to be signed, the simulator generates tuples by computing $T = G(M)$, then picking U at random between 0 and $|\mathcal{H}| - 1$ and S at random between 0 and $q - 1$. The simulator computes $E_G = F_2(S, T, U)$, $E_Y = F_3(S, T, U)$ and $R = g^{E_G} Y^{E_Y} \bmod p$. It then defines $H(R)$ to be equal to U . In the event that $H(R)$ was already defined, the simulator would declare failure. These tuples will be uniformly distributed among all verifiable tuples $(R, S, T = G(M), U)$ such that $R = g^{E_G} Y^{E_Y} \bmod p$.

The adversary can only distinguish between this distribution and the signer's one if the simulator computes an R for which $H(R)$ was already defined or if the signer computes an R which he had used earlier. Let b denote the number of queries that have been made to the random oracle H . The probability of one of these events happening is less than $1 - e^{-b(b-1)/2q}$ (birthday paradox) which can be approximated by $b^2/2q$. □

Theorem 12. *Suppose that H is an ideal random function but G a collision-resistant hash function. Given an attacker \mathcal{A} that can find with probability ε a verifiable tuple (M, R, S, T, U) for a new message M , with less than Q queries to the hash function H , then with constant probability $1/96$, with less than $25Q/\varepsilon$ replays of \mathcal{A} , with different random oracles, \mathcal{A} extracts the secret key X .*

Proof. When the attacker \mathcal{A} outputs a new verifiable tuple (M, R, S, T, U) , either $H(R)$ had been defined by the simulator (case 1) or directly by H (case 2).

- case 1: the simulator had produced a verifiable tuple, (M', R, S', T', U') , for which $M \neq M'$ and therefore $T = G(M) \neq G(M') = T'$, since G is collision-resistant. Because of the TEGTSS-I properties, one has two distinct representations of the same R in the basis (g, Y) , which leads to X [6].
- case 2: \mathcal{A} outputs a verifiable tuple, (M, R, S, T, U) , in which $R = Q_j$ for some $j \leq Q$ and \mathcal{A} made a direct query for the value of $H(Q_j)$. Using the forking lemma (Lemma 10), after less than $(1 + 24Q)/\varepsilon$ replays of \mathcal{A} , one gets two tuples (M_1, R, S_1, T_1, U_1) and (M_2, R, S_2, T_2, U_2) such that $U_1 \neq U_2$. With a closer look at the proof of the forking lemma, one can see that U_2 follows the uniform distribution. Given U_2 , let T_2 be the unique value such that $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$ for a verifiable tuple. By the assumption that G is collision-resistant, and therefore one-way, the probability that \mathcal{A} can find a message M_2 such that $G(M_2) = T_2$ is vanishingly small. Consequently, we likely have $F_3(S_1, T_1, U_1) \neq F_3(S_2, T_2, U_2)$ and thus two distinct representations of the same R in the basis (g, Y) , which leads to X [6].

□

5.3 TEGTSS – Type II

Let us now study the Type II (which includes the DSA-II scheme), proving first the existence of an indistinguishable simulator.

Lemma 13. *Suppose H is an ideal random function with output between 0 and $|\mathcal{H}|-1$. Then there exists a simulator that creates verifiable tuples such that after b steps, the probability that the simulator can be distinguished from a signer is less than $b^2/2q$.*

Proof. Given a message M to be signed, the simulator generates tuples by first picking A and B at random, both in \mathbb{Z}_q . It then computes $R = g^A Y^B \bmod p$ and $T = G(R)$. Using the property of F_2 and F_3 , S and U are defined as the only values leading to both $E_G = A$ and $E_Y = B$. Then $H(M, T)$ is defined to be equal to U . As above, this simulation is indistinguishable but with an advantage upper-bounded by $b^2/2q$. □

Theorem 14. *Let us assume that H is an ideal random function and G an $(\ell+1)$ -collision-resistant function. Given an attacker \mathcal{A} that can find with probability ε a verifiable tuple (M, R, S, T, U) for a new message M , with less than Q queries to the hash function H , then with constant probability $1/96$, with less than $25Q\ell \log(2\ell)/\varepsilon$ replays of \mathcal{A} , with different random oracles, \mathcal{A} extracts the secret key X .*

Proof. \mathcal{A} outputs a verifiable tuple, (M, R, S, T, U) , in which $(M, T) = Q_j$ for some $j \leq Q$ and \mathcal{A} made a direct query for the value of $H(Q_j)$, since M is a new message, never asked of the simulator. Using the forking lemma

(Lemma 10), after less than $(1 + 24Q\ell \log(2\ell))/\varepsilon$ replays of \mathcal{A} , one gets $\ell + 1$ tuples (M, R_i, S_i, T, U_i) such that the U_i are pairwise distinct, with $T = G(R_i)$.

Since G is $(\ell + 1)$ -collision-resistant, there exists a pair of indices (i, j) for which we have $R_i = R_j$. Assume that $E_{G_i} = E_{G_j}$ and $E_{Y_i} = E_{Y_j}$. Then, because of the TEGTSS-II properties, $S_i = S_j$ and $U_i = U_j$, which contradicts the fact that the U_i are all distinct. Then from two representations of the same R in basis (g, Y) , one gets X [6]. \square

Theorem 15. *Let us assume that H is an ideal random function and $x \mapsto G(g^x \bmod p)$ an $(\ell + 1)$ -collision-free function. Given an attacker \mathcal{A} that can find with probability ε a verifiable tuple (M, R, S, T, U) for a new message M , with less than Q queries to the hash function H , then with constant probability $1/96$, with less than $25Q\ell \log(2\ell)/\varepsilon$ replays of \mathcal{A} , with different random oracles, \mathcal{A} extracts the secret key X .*

Proof. As above, after less than $(1 + 24Q\ell \log(2\ell))/\varepsilon$ replays of \mathcal{A} , one gets $\ell + 1$ tuples (M, R_i, S_i, T, U_i) such that the U_i are pairwise distinct, with $T = G(R_i)$ and $R_i = g^{E_{G_i}} Y^{E_{Y_i}} = g^{x_i} \bmod p$ for some x_i . Since $x \mapsto G(g^x \bmod p)$ is $(\ell + 1)$ -collision-free, the same conclusion as above holds. \square

6 Application to Some Signature Schemes

Lemma 16. *If G is just collision-resistant but H a random oracle, the KCDSA is unforgeable relative to the discrete logarithm problem.*

Proof. It is an immediate corollary from Lemma 6 and Theorem 8. \square

6.1 The DSA-II Variant

Lemma 17. *If G is an $(\ell + 1)$ -collision-resistant function or $x \mapsto G(g^x \bmod p)$ is an $(\ell + 1)$ -multi-collision-free function, but H a random oracle, then DSA-II is unforgeable relative to the discrete logarithm problem.*

Proof. It is an immediate corollary from Lemma 7 and Theorem 8. \square

Remark 1. One can first remark that for any random function G , the probability that $x \mapsto G(g^x \bmod p)$ has a $(\ell + 1)$ -multi-collision is approximately less than $q/(\ell + 1)!$ [24], which is very small for $\ell = \log q$.

This provides a *security argument* for a very slight variant of the original DSA, where the $H(M)$ is just replaced by $H(M, T)$. Indeed, it is very unlikely that the “ $x \mapsto (g^x \bmod p) \bmod q$ ” map has $(\log q)$ -multi-collision. Indeed, even just a 2-collision would lead to an important weakness in the original DSA design by an attack similar than Vaudenay’s [31]: if for a given (p, q, g) provided by a honest authority someone happens to find out a 2-collision

$$T = g^k \bmod p \bmod q = g^{k'} \bmod p \bmod q$$

then he can choose two different messages M and M' and a particular X as his private key so that the signatures of M and M' collide. Namely, if

$$X = \frac{kH(M') - k'H(M)}{T(k - k')} \bmod q$$

then $(H(M) + XT)/k \bmod q = S = (H(M') + XT)/k' \bmod q$ so that he can reveal the signature (S, T) of M and later on claim it was the signature of M' .

6.2 Short-Length Signatures

The Type II of TEGTSS has the attractive property of providing short signatures. Indeed, a hash function that returns 80-bits digests can be considered as 5-collision-resistant, since a search would require 2^{64} computations [13]. Therefore, since the practical signature consists of the pair (S, T) where $S \in \mathbb{Z}_q$ and T the digest produced by G , a 5-collision-resistant hash function, it can be shorter than 30 byte-long.

7 Conclusion

We have studied and validated security (under the random oracle model) of general schemes which include some standardized schemes or very close variant thereof. We proved their security while maintaining the efficiency of the standard schemes. We, therefore, believe that perhaps the standard bodies should look carefully into our study.

References

1. R. Anderson and S. Vaudenay. Minding your p 's and q 's. In *Asiacrypt '96*, LNCS 1163, pages 26–35. Springer-Verlag, Berlin, 1996. [280](#)
2. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCCS*, pages 62–73. ACM Press, New York, 1993. [277](#), [278](#), [279](#), [281](#)
3. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995. [279](#), [281](#)
4. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996. [278](#), [279](#), [281](#)
5. D. Bleichenbacher. Generating El Gamal Signatures without Knowing the Secret Key. In *Eurocrypt '96*, LNCS 1070, pages 10–18. Springer-Verlag, Berlin, 1996. [280](#)
6. S. A. Brands. An Efficient Off-Line Electronic Cash System Based on the Representation Problem. Technical Report CS-R9323, CWI, Amsterdam, 1993. [288](#), [289](#)
7. E. F. Brickell. Invited lecture given at the Crypto '96 conference. unpublished manuscript. [278](#), [281](#)

8. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218. ACM Press, New York, 1998. [277](#)
9. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976. [276](#)
10. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985. [277](#), [280](#)
11. U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, 1:77–95, 1988. [285](#)
12. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto ’86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987. [278](#)
13. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto ’94*, LNCS 839, pages 202–215. Springer-Verlag, Berlin, 1994. [290](#)
14. S. Goldwasser, S. Micali, and R. Rivest. A “Paradoxical” Solution to the Signature Problem. In *Proc. of the 25th FOCS*, pages 441–448. IEEE, New York, 1984. [277](#)
15. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988. [277](#), [278](#), [279](#)
16. ISO. ISO/IEC 14888 Final Draft – Information Technology – Security Techniques - Digital Signatures with Appendix. International Organization for Standardization, Berlin, Germany, 1998. [281](#)
17. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. IEEE P1363a Submission, August 1998.
Available from <http://grouper.ieee.org/groups/1363/addendum.html>. [277](#), [280](#)
18. C. H. Lim and P.J. Lee. A Study on the Proposed Korean Digital Signature Algorithm. In *Asiacrypt ’98*, LNCS 1514, pages 175–186. Springer-Verlag, Berlin, 1998.
19. M. Naor and M. Yung. Universal One-way Hash Functions and their Cryptographic Applications. Proceedings of 21st STOC, May 1989. [277](#)
20. NIST. *Digital Signature Standard* (DSS). Federal Information Processing Standards Publication 186, November 1994. [277](#), [280](#)
21. NIST. *Secure Hash Standard* (SHS). Federal Information Processing Standards Publication 180–1, April 1995. [279](#), [280](#)
22. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Crypto ’98*, LNCS 1462, pages 354–369. Springer-Verlag, Berlin, 1998. [278](#), [279](#), [280](#), [281](#), [285](#)
23. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 1999.
Available from <http://www.di.ens.fr/~pointche>. [278](#), [279](#), [280](#), [281](#), [284](#), [285](#)
24. D. Pointcheval and S. Vaudenay. On Provable Security for Digital Signature Algorithms. Technical Report LIENS-96-17, LIENS, October 1996. [278](#), [281](#), [282](#), [289](#)
25. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, The Internet Engineering Task Force, April 1992. [279](#)
26. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. [278](#)

27. J. Rompel. One-way Functions are Necessary and Sufficient for Signature. Proceedings of 22d STOC, May 1990. [277](#)
28. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990. [278](#), [280](#)
29. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991. [278](#), [280](#)
30. P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Eurocrypt '96*, LNCS 1070, pages 332–343. Springer-Verlag, Berlin, 1996. [280](#)
31. S. Vaudenay. Hidden Collisions on DSS. In *Crypto '96*, LNCS 1109, pages 83–88. Springer-Verlag, Berlin, 1996. [289](#)

Optimally Efficient Accountable Time-Stamping

Ahto Buldas^{*1}, Helger Lipmaa^{*1}, and Berry Schoenmakers^{**2}

¹ Küberneetika AS, Akadeemia tee 21, 12618 Tallinn, Estonia
{ahtbu,helger}@cyber.ee

² Dept. of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
berry@win.tue.nl

Abstract. Efficient secure time-stamping schemes employ a 2-level approach in which the time-stamping service operates in rounds. We say that a time-stamping service is *accountable* if it makes the TSA and other authorities accountable for their actions by enabling a principal to detect and later prove to a judge any frauds, including attempts to reorder time-stamps from the *same* round. We investigate the paradigm of time-stamping services based on simply connected graphs, and propose a simple, yet optimal, accountable time-stamping service, using what we call threaded tree schemes. We improve upon the previously best scheme by Buldas and Laud by reducing the size of a time stamp by a factor of about 3.786 and show that our construction is optimal in a strict sense. The new protocols also increase the trustworthiness of the publication process, which takes place at the end of each round.

1 Introduction

A time-stamping service is a collection of protocols providing long-term authentication of digital documents together with the moment in time at which they were submitted for authentication. Time-stamping is expected to become an integral part of the legal framework for digital documents, as it enables one to prove in court that a document existed at some given point in time. In addition, time-stamping helps to significantly lower the level of trust currently required of a public-key infrastructure by making it possible to prove that a document was signed before the revocation act of the corresponding signature key was stamped. As such, one will frequently depend on time-stamping to resolve the status of documents and corresponding signature keys, which motivates the need for a clear understanding of the security and efficiency requirements of time-stamping.

In the context of time-stamping it is common to regard time as a relative notion. We say that a (time-)stamping service provides *relative temporal authentication* if one is able to decide which stamp has been issued first for each pair

^{*} Supported partially by Estonian Science Foundation grant 3742.

^{**} Part of this work was done while visiting Küberneetika AS.

of time-stamps. Following Haber and Stornetta [HS90, HS91], a cryptographic way to achieve such a relative order for the documents is to create dependencies between the time certificates attached to the documents, where later certificates are obtained by applying a collision-resistant function to earlier ones. A short description of the existing time-stamping schemes is given in Section 3.

We consider time-stamping services that operate in rounds, where clients will issue requests to a central time-stamping authority (TSA) which in turn hands over the cumulative round stamp to the publication authority (PA) so the latter may publish it. In order that the time-stamps are meaningful in court, it is required that one is able to actually *prove* that a relative temporal ordering holds for any pair of time-stamps (even for time-stamps from the same round), or to prove that an authority deviated from the protocols (be it accidentally or intentionally). It is important to note that no single player in the scheme (including the TSA and PA) is required to store all of the time stamps for the rounds in which it takes part.

A time-stamping service is *accountable*, if it makes the TSA and other authorities accountable for their actions by enabling a principal to detect and later prove to a judge any frauds, including attempts to reorder time-stamps from the same round. Moreover, if a honest party followed the protocol but is still accused of forgeries, she can explicitly disavow any false accusations. Section 4 addresses the security objectives in more depth, where we will rely on notions such as simply connected graphs and authentication graphs, which are introduced in Section 2.

In Sections 5–7 we present our main contributions. In Section 5 we introduce a notion of general ϕ -schemes, based on simply connected graphs. In particular, we define two instances of it: anti-monotone schemes [BLV98, BL98], and our *threaded tree schemes*. We show that while the latter scheme is not anti-monotone it satisfies the same security objectives as achieved by anti-monotone schemes. The important difference is that the size of the time certificates becomes approximately 3.786 times shorter than for the optimal anti-monotone schemes of [BL98], and overall the scheme also becomes simpler.

Section 6 presents an optimized set of time-stamping protocols using simply connected graphs. As a novel feature, we present a *publication protocol*, in addition to the *stamping protocol* and the *stamp completion protocol* as described for other time-stamping services. Finally, in Section 7, we prove that under general assumptions our scheme based on threaded tree schemes is tightly optimal (i.e., optimality is not just asymptotic).

2 Preliminaries

2.1 Cryptographic Primitives

The basic cryptographic tools used in a time-stamping service are hash functions and digital signatures. We let $h : \{0, 1\}^* \rightarrow \{0, 1\}^k$ denote some fixed *collision-resistant hash function*, where k is a security parameter. Furthermore, we let

$\text{sig}_A(M)$ denote A 's signature on message M . Since during stamping protocols the principals must sign arbitrary data, we require the signature scheme to be existentially unforgeable against adaptive chosen message attack.

2.2 Authentication Graphs

Let $G = (V, E)$ be a directed acyclic graph such that $V = \{1, 2, \dots, |V|\}$ and $v < w$ whenever $(v, w) \in E$, i.e., the vertices of G are topologically sorted (since G is a directed acyclic graph, such a numbering always exists). For any $W \subseteq V$ we define $E(W) = \{v : (\exists w \in W)(w, v) \in E\}$ and $E^{-1}(W) = \{v : (\exists w \in W)(v, w) \in E\}$. We assume that $v = |V|$ is the unique vertex with $E(\{v\}) = \emptyset$, which we call the *root* of G . Finally, we let $\mathcal{S}(G) = \{v \in V : E^{-1}(\{v\}) = \emptyset\}$ be the set of *source* vertices of G .

Graph G is called *simply connected* if there is a directed path between each pair of vertices v and w (going either from v to w or from w to v). For topologically sorted G it is equivalent to state that $(v, v+1) \in E$ for all v , $1 \leq v < |V|$. For each pair $v < w$ of vertices of a simply connected graph G we define the *distance* $d(v, w)$ as the length of the shortest path from v to w .

We say that $U \subseteq V$ is *computable* from $W \subseteq V$ if either (1) $U \subseteq W$ or (2) $U \cap \mathcal{S}(G) \subseteq W$ and $E^{-1}(U \setminus W)$ is computable from W . For any subset $W \subseteq V \setminus \mathcal{S}(G)$, we say that the set $\mathbf{A}_G(W) = E^{-1}(W) \setminus W$ is the *authenticator* of W . Clearly, W is computable from $\mathbf{A}_G(W)$.

Recall that h denotes a collision-resistant hash function. As a generalization of Merkle's authentication trees [Mer80], we introduce *authentication graphs* G as labeled directed acyclic graphs with the labels assigned in the following manner. Each source v of G is labeled by $L_v = H_v$, where H_v is a given string (data element) specific to source v . The label of a non-source vertex v is computed as a function of the labels of its predecessors: $L_v = h(\mathbf{L}_{E^{-1}(v)})$. Here $\mathbf{L}_S = (L_{v_1}, \dots, L_{v_k})$, where v_1, \dots, v_k are the elements of S in strictly increasing order. We often identify the labels of an authentication graph with its vertices and say that a vertex is computed from its predecessors rather than the label of a vertex is computed from the labels of its predecessors.

3 State of the Art

Modern time-stamping services provide *relative temporal authentication*, where one verifies the relative order of stamp issuing, given any two time stamps. In all such schemes the time certificate of a later issued stamp is dependent on earlier stamps (a paradigm first proposed in [HS90]).

Most of the proposed schemes are built upon authentication graphs. *Linear linking schemes* [HS90] use a chain as the underlying graph. This solution is clearly impractical, since certificate length increases linear with the number of stamps issued so far. In the worst case, verification of a single time certificate takes as much time as it takes the TSA to compute the stamps for all documents issued so far.

Tree schemes [BdM91,BHS92] (using a tree as the underlying graph) were motivated by the desire to minimize the storage and communication requirements. Since trees are not simply connected, this type of schemes do not provide relative temporal authentication within rounds. To overcome this, the stamping procedure was divided into rounds, so that the cumulative round stamp of every round was published in an authenticated medium. Additionally, the duration of rounds was assumed to be small enough to think about the stamping requests submitted during the same round as simultaneous. Unfortunately, bounding the round lengths reduces the compression effect.

Anti-monotone schemes [BLLV98, BL98] combine the simply connectedness of linear schemes and efficiency of tree schemes, therefore sharing the best features of these schemes. Anti-monotonicity makes the schemes slightly less efficient (and also, more difficult to understand and implement) than the tree schemes.

Accumulator schemes [BdM93] (the only known viable alternative to the graph based schemes) further reduce the storage requirements, by introducing a trapdoor into the scheme. However, accumulator schemes share all drawbacks of tree schemes. They have also another principal weakness: one has to trust the coalition of parties who generated the trapdoor information. Hence, usage of such schemes would directly violate the objective of reducing trust in stamping services. To date, there is no known efficient construction of trapdoorless accumulators (see [San99] for recent work in this area).

4 Time-Stamping Objectives

A *stamping service* consists of a set of principals with the time-stamping authority (TSA) and the publication authority (PA) together with a quadruple (S, C, V, P) of protocols. The *stamping protocol* S is used by a participant to hand over a message to the TSA for time-stamping. During the *stamp completion protocol* C a participant obtains a time certificate from the TSA. The *verification algorithm* V is used by a principal having two time certificates to verify the temporal order of the corresponding stamping events. The *publication protocol* P is used by a TSA to handle the round stamp (short fingerprint of the round) to the PA who will publish it on some authenticated and easily accessible medium.

We say a time-stamping service is *accountable* if the next two properties hold whenever there is at least one uncorrupted party (say, one honest client interested in legal value of a particular time stamp) during the creation of stamped information:

- **Fraud detection.** The service makes the trusted third parties accountable for their actions by enabling a principal to detect and later prove to a judge any frauds affecting the relative ordering between time-stamps.
- **Anti-framing.** If a party has honestly followed the protocol but is still accused in forgeries, she can explicitly disavow any false accusations.

One of the crucial requirements of an accountable time-stamping service is to withstand *reordering attacks*, where the TSA assigns an earlier stamp to a document submitted later than another document, i.e., they should provide *relative temporal authentication*. While network latency makes it impossible to detect microscale reordering attacks, it should be intractable for the stamping service to forge the order of time stamps for documents submitted in significantly different time moments. An ideal time-stamping service should provide relative temporal authentication between any pair of documents.

Simple time-stamping services (like hash-and-sign and simple tree schemes) not ensuring relative ordering of time stamps within rounds are highly vulnerable to reordering attacks. However, these attacks can partially be eliminated by using the stamping protocols defined in [BLLV98], where directly after the v th stamping request H_v the TSA returns a signed receipt on some value $L_v = h(H_1, \dots, H_v)$ to the client, where h is a collision-resistant hash function (informally, we say then that L_v is *(one-way) dependent on* H_i , $i = 1, \dots, v$). If L_v is additionally dependent on L_i , $1 \leq i < v$, the client can use TSA's signatures on the values L_v to prove any frauds.

Unfortunately, this methodology helps only until TSA's signature can be trusted (in the worst case, only until the end of the current round). If all L_v were not authenticated by the round stamp, there would be no way to detect the frauds after TSA's key becomes invalid. Hence, a time-stamping service is accountable only if the round stamp authenticates all the values L_v , where every L_v is dependent on L_{v-1} . Moreover, every L_v should be dependent on the round stamp of the previous round. The resulting structure can be seen as a *dependency graph* that has the round stamp as a label of its root and as a subgraph a simply connected graph with vertices labeled by values L_v , where the edges correspond to applying a collision-resistant hash function, and the sources correspond to the data elements H_v . For efficiency we assume that each L_v is a k -bit output of a fixed hash function h . We additionally assume that L_1 is equal to the stamp for the previous round, since the rounds have to be connected. A time certificate has to contain sufficient data to prove that the mentioned dependencies hold. We shall make these informal notions precise in the next sections.

It is important to realize what time-stamping does and what it does not. For example, there is a type of reordering attack that is often used as an argument against linking schemes. Let H_1, \dots, H_n be the stamping requests during a round. Before "officially" closing the round, the TSA may issue additional time stamps for H_n, H_{n-1}, \dots, H_1 in reverse order. After that the TSA is able, for any pair of stamps of H_i and H_j , to present proofs for the statements " H_i was stamped before H_j " and " H_j was stamped before H_i ". It may seem, therefore, that using linking schemes and signed receipts does not give any advantage compared to simple hash-and-sign scheme.

The following example shows a weakness in this argument (cf. [BLLV98, Section 3.1]). First, a client requests a stamp L_n of a nonce from the TSA. Subsequently, she stamps her signature σ on the actual document X and L_n . The TSA may then issue additional stamps for X and σ . However, no one is able

to back-date new documents relative to σ without cooperating with the signer. Note that this attack is hard to prevent if the underlying graph is not simply connected, and that usually the TSA is trusted not to mount such an attack.

5 Schemes Based on Simply Connected Graphs

5.1 General Construction

The first efficient accountable time-stamping service proposed in [BLV98] is based on *anti-monotone* schemes. Briefly, the intuition behind this scheme is to define the time certificates in such a way that for any v and w , $v < w$, the union of the v th and w th time certificates *contains* set-theoretically a dependent path between the corresponding stamps L_v and L_w . In what follows we show that up to 3.786 times shorter certificates can be achieved if we instead assume that said union contains only information sufficient to *compute* this path. For this we shall define a general framework that includes in particular anti-monotone schemes and the new optimal threaded authentication schemes.

Let $G = (V, E)$ be a rooted simply connected graph with a single source. Let $\phi : V \rightarrow \{\text{red, black}\}$ be an arbitrary function, such that $\phi(1) = \text{red}$. We regard ϕ as a red/black-coloring of G . The ϕ -scheme $\llbracket G \rrbracket_\phi$ is constructed from G by adding individual data elements, that is, a new vertex v_ϕ and an edge (v_ϕ, v) , to every black vertex v . Then we say that v_ϕ is a *data element connected to vertex* ϕ . We say that $\mathcal{P} : V \rightarrow 2^V$ is a *pass-through function* if it maps an arbitrary vertex v to a path $(w, \dots, v, \dots, |V|)$, such that $(1, w) \in E$.

Definition 1. Let $\llbracket G \rrbracket_\phi$ be a ϕ -scheme, where $G = (V, E)$, and let $\mathcal{P} : V \rightarrow 2^V$ be a pass-through function. Now, let $v \in V$ be a black vertex. We define $\text{head}(v)$ to be the set of vertices in $A_{\llbracket G \rrbracket_\phi}(\mathcal{P}(v))$ that can be computed from the set of data elements connected to black vertices $w \leq v$, and we set $\text{tail}(v) = A_{\llbracket G \rrbracket_\phi}(\mathcal{P}(v)) \setminus \text{head}(v)$. The time certificate $\text{Cert}(v)$ of a source $v \in S(\llbracket G \rrbracket_\phi)$ is defined as $\text{Cert}(v) = (v, \mathbf{L}_{\text{head}(v)}, \mathbf{L}_{\text{tail}(v)})$.

Recall that every source v of an authentication graph G corresponds to some string H_v . Intuitively, $\text{Cert}(v)$ consists of the minimal amount of data necessary to verify whether v lies on a path between 1 (for which the label is equal to the last round stamp) and $|V|$ (for which the label is equal to the current round stamp). If v is represented by k bits then $|\text{Cert}(v)| = k(|A_{\llbracket G \rrbracket_\phi}(\mathcal{P}(v))| + 1)$.

Jumping ahead a little, in the time-stamping protocols $\text{head}(v)$ corresponds to the maximum subset of $A_{\llbracket G \rrbracket_\phi}(\mathcal{P}(v))$ that the stamping authority knows directly after stamping the v th element and $\text{tail}(v)$ corresponds to the remainder of the authenticator. Since $\mathbf{L}_{\text{head}(v)}$ is dependent on all data elements issued thus far, by returning his signature on $\mathbf{L}_{\text{head}(v)}$ (or on some value that is one-way dependent on $\mathbf{L}_{\text{head}(v)}$) to a client directly after issuing the v th stamp, the stamping authority commits himself to all data elements stamped before the v th element. Transmission of the certificate to a client at the end of the round commits the authority to every data element of this round even if the authority's signing key gets compromised later on. (The time-stamping protocols are given in Section 6.)

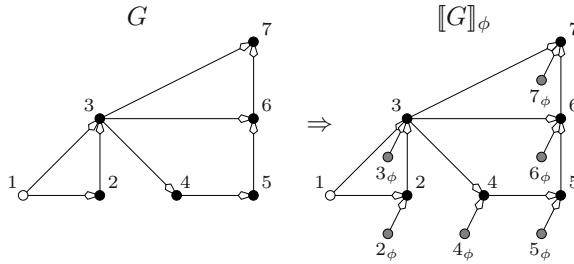


Fig. 1. Construction of $\llbracket G \rrbracket_\phi$ for anti-monotone G .

5.2 Anti-Monotone Schemes

Next we show that even if using exactly the same base graphs as in [BLLV98] one can get a significant improvement over their solution. A simply connected graph $G = (V, E)$ is *anti-monotone* if the in-degree of each vertex is at most 2, and if for all $(v_1, w_1), (v_2, w_2) \in E$ we have that $v_1 < w_2 < w_1$ implies $v_1 \leq v_2$. Let $\phi : V \rightarrow \{\text{red, black}\}$ be such that $\phi(v) = \text{black}$ iff $v \neq 1$. The *anti-monotone* ϕ -scheme $\llbracket G \rrbracket_\phi$ is a ϕ -scheme constructed from anti-monotone graph G .

As shown in [BLLV98], in this case $\text{tail}(v) \cap \text{head}(w)$ is nonempty for any $v < w$. Since $\text{Cert}(v)$ (respectively $\text{Cert}(w)$) contains by definition information necessary to compute all the elements in $L_{\text{tail}(v)}$ (respectively in $L_{\text{head}(w)}$), then for any $u \in \text{tail}(v) \cap \text{head}(w)$, the union of $\text{Cert}(v)$ and $\text{Cert}(w)$ contains sufficient information to verify that L_u is dependent on L_v and L_w is dependent on L_u .

Let $d_{pt}(G) = \max_{v \in V} (d(1, v) + d(v, |V|))$. Buldas and Laud proved in [BL98] that for any anti-monotone graph G , $d_{pt}(G) \geq 1.893 \log_2 |\mathcal{S}(G)| + O(1)$. Using the ϕ -scheme $\llbracket G \rrbracket_\phi$ we get that the length of a certificate $\text{Cert}(v)$ is equal to $\log_2 |\mathcal{S}(G)| + |\text{head}(v)| + |\text{tail}(v)|$, and hence, $\max_v |\text{Cert}(v)| \geq (k + o(1)) \cdot d_{pt}(G)$. However, the binary linking schemes of [BLLV98, BL98] used redundant certificates such that $\max_v |\text{Cert}(v)| \geq (2k + o(1)) \cdot d_{pt}(G)$ and hence casting the problem in a more general view has helped to reduce the length of the certificates by a factor of two.

5.3 Threaded Authentication Trees

The security of the last construction depends only on the property that there is a vertex u that can be computed from the union of $\text{Cert}(v)$ and $\text{Cert}(w)$. As such element always exists (take the root), the anti-monotonicity property is not necessary for the stamping service to be accountable. Below we present a new construction based on *threaded authentication trees*. Optimality of this scheme in the general case will be proven later.

For Merkle's authentication tree T_d of depth d , the certificate length for a vertex v is $k(d + 2)$, where k is the size of the outputs of hash function h . In their stamping system, Benaloh and de Mare [BdM93] added a new vertex

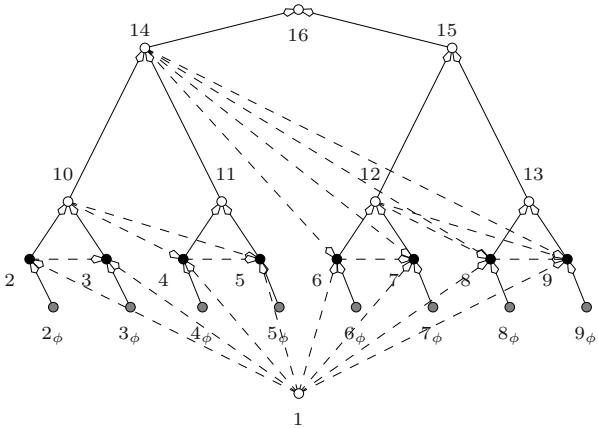


Fig. 2. Threaded tree scheme $\llbracket W_3 \rrbracket_\phi$.

1 with edges $(1, w)$, for any $w \in \mathcal{S}(T_d)$, to Merkle's authentication tree. The resulting graph has certificate length $k(d + 3)$, and the v th certificate contains dependency from R_{r-1} to L_v to R_r . We will proceed further by adding edges to the Benaloh-de Mare scheme in such a way that the resulting graph will be simply connected, but *without* enlarging the certificates. The method presented below is not unique in this respect, but has the advantage that it permits an efficient implementation.

Let $T_d = (V, E)$ be the complete binary tree of depth d . We number the vertices of T_d level by level, starting at the lowest level (at depth d) and from left to right so that the leaves get numbers $2, \dots, 2^d + 1$, and the root gets number $|V| = 2^{d+1}$. Let $\phi : V \rightarrow \{\text{red, black}\}$ color the sources of T_d (i.e., vertices $2 \leq v \leq 2^d + 1$) black and inner nodes red.

We define the *threaded authentication tree* W_d as the graph built from T_d by applying the next two steps:

1. for each $v \in \mathcal{S}(T_d)$ consider the unique root path $\mathcal{P}(v)$ starting from v and add an edge (w, v) for each left child w of a vertex on $\mathcal{P}(v)$ provided $w \notin \mathcal{P}(v)$, and
2. add a vertex 1 and, for each $v \in \mathcal{S}(T_d)$, add an edge $(1, v)$. Vertex 1 is labeled by $L_1 = R_{r-1}$, where r is number of the current round. We color 1 red, since it corresponds to a data element.

Note that the set of vertices w for which the edge (w, v) was added in the step 1 is equal to $\text{head}(v)$. We define the d th *threaded tree scheme* to be the ϕ -scheme $\llbracket W_d \rrbracket_\phi$. Clearly, a threaded authentication tree W_d is simply connected (consider a postorder traversal of the binary tree T_d), but it is not anti-monotone.

As an example, consider the threaded tree scheme $\llbracket W_3 \rrbracket_\phi$ of Figure 2. Graph W_3 is simply connected because it contains the postorder traversal $2 \rightarrow 3 \rightarrow 10 \rightarrow 4 \rightarrow 5 \rightarrow 11 \rightarrow 14 \rightarrow 6 \rightarrow 7 \rightarrow 12 \rightarrow 8 \rightarrow 9 \rightarrow 13 \rightarrow 15 \rightarrow 16$ as a directed path. Then

INPUT: $(C, L_v, R_{r-1}, R_r, H_v)$, where $C = (v, \chi, \tau)$, $\chi = (\chi_1, \dots, \chi_m)$, $\tau = (\tau_1, \dots, \tau_n)$.
 Reject if $\chi_1 \neq R_{r-1}$ or $\chi_2 \neq H_v$ or $\tau_n \neq R_r$.
 Reject if $L_v \neq h(\chi)$.
 $\ell := L_v$, $p := 1$; $q := 3$.
 For $i := 0$ to $\log_2 v$ do:
 if $(v - 1) \& 2^i = 0$ then $\ell := h(\ell, \tau_p)$; $p := p + 1$ else $\ell := h(\chi_q, \ell)$; $q := q + 1$.
 Reject if $\ell \neq R_r$. Otherwise accept.

Fig. 3. Certificate verification algorithm VfyCert for threaded tree schemes.

$\text{head}(4) = (1, 4_\phi, 10)$, $\text{tail}(4) = (5, 15)$, $\text{head}(8) = (1, 8_\phi, 12, 14)$, $\text{tail}(8) = (9)$, $\text{Cert}(4) = (4; L_1, L_{4_\phi}, L_{10}; L_5, L_{15})$ and $\text{Cert}(8) = (8; L_1, L_{8_\phi}, L_{12}, L_{14}; L_9)$. The set $\text{Cert}(4) \cup \text{Cert}(8)$ contains sufficient information to first compute and then verify the following path from R_{r-1} to L_4 to L_8 to R_r , where r is the number of the current round:

$$\begin{aligned} \underbrace{\mathbf{R}_{r-1} = L_1}_{\text{head}(4) \cap \text{head}(8)} &\rightarrow \mathbf{L}_4 = h(\underbrace{R_{r-1}, L_{4_\phi}, L_{10}}_{\text{head}(4)}) \rightarrow L_{11} = h(L_4, \underbrace{L_5}_{\text{tail}(4)}) \rightarrow \\ L_{14} &= h(\underbrace{L_{10}}_{\text{head}(4)}, L_{11}) \rightarrow \mathbf{L}_8 = h(\underbrace{R_{r-1}, L_{8_\phi}, L_{12}, L_{14}}_{\text{head}(8)}) \rightarrow \\ L_{13} &= h(L_8, \underbrace{L_9}_{\text{tail}(8)}) \rightarrow L_{15} = h(\underbrace{L_{12}}_{\text{head}(8)}, L_{13}) \rightarrow L_{16} = h(\underbrace{L_{14}}_{\text{head}(8)}, \underbrace{L_{15}}_{\text{tail}(4)}) = \mathbf{R}_r . \end{aligned}$$

Certificate verification algorithm VfyCert (which is used in the verification algorithm) is shown in Figure 3.

For any vertices $v, w \in W_d$, $v \leq w \Rightarrow d(v, w) \leq 2d - 1$. Thus, the verification can be done very efficiently, and $|\text{Cert}(v)| = k(d + 3)$, for all v . It will be proven in Section 7 that this is also the lower bound. While in the case of schemes used in [BLLV98, BL98], the union of two time certificates contained the whole “proof” of dependency, in the current case the proof itself is not contained in, but can still be computed from the union. That is, intuitively, the main source of the redundancy in the anti-monotone schemes compared to the new scheme. We think that this result is interesting in its own right in graph theory.

6 Protocols

As mentioned in Section 4, protocols are crucial for the overall security of a time-stamping service. In this section we briefly describe the three protocols constituting our time-stamping service. The stamping protocol and the stamp completion protocol are improvements over the protocols of [BLLV98, Section 4.2], using threaded authentication trees. The publication protocol is an additional protocol required in our model of time-stamping. See Figure 4 for an overview of the protocols. We assume the base graph G and the functions \mathcal{P} and ϕ to be fixed.

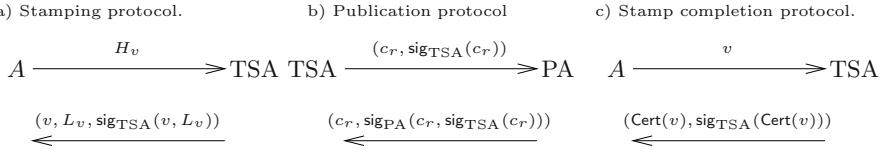


Fig. 4. Protocols of a time-stamping service.

The following algorithm is used in the stamping protocols to verify whether $v_1, v_2 \in \mathcal{S}(\llbracket G \rrbracket_\phi)$ and $v_1 < v_2$ (i.e., whether the corresponding documents H_{v_1} and H_{v_2} were stamped during the same round and further that H_{v_1} was stamped before H_{v_2} was). Note that the algorithm proposed in [BLLV98] included an unnecessary checking whether $\text{tail}(v_1) \cap \text{head}(v_2) \neq \emptyset$ as a separate step.

VERIFICATION ALGORITHM. Let $C = (v, \mathbf{L}_\chi, \mathbf{L}_\tau)$, where v is a non-source vertex and χ and τ are paths in $\llbracket G \rrbracket_\phi$, such that v is computable from χ and $|V|$ is computable from τ . Let $\text{VfyCert}(C, L_v, L_{|V|})$ be a function that accepts iff χ is authenticator of some path from 1_ϕ to v and τ is authenticator of some path from v_ϕ to $|V|$, and the value of a candidate v computed from the values in \mathbf{L}_χ is equal to v (the same verification is done for $|V|$ and \mathbf{L}_τ). The algorithm V gets as input a tuple $(C_1, C_2, L_{v_1}, L_{v_2}, R_{r-1}, R_r, H_v)$, where $C_i = (v_i, \chi_i, \tau_i)$ and accepts iff (1) $\text{VfyCert}(C_i, L_{v_i}, R_{r-1}, R_r, H_v)$ holds, for $i \in \{1, 2\}$; and (2) $v_1 < v_2$.

STAMPING PROTOCOL. See Fig. 4(a). The client A begins the protocol by sending hash H_v of the v th document to the TSA. The TSA calculates $L_v = h(\mathbf{L}_{\text{head}(v)})$, adds L_v to the database of stamps, and sends A the second message. The client checks if the message is of the right form (note that he cannot yet verify whether L_v was computed correctly).

PUBLICATION PROTOCOL. See Fig. 4(b). Here $c_r = (L_{|V|}, r)$ and $R_r = L_{|V|}$ is the cumulative round stamp of the r th round. After the end of the r th round the TSA begins the publication protocol by sending the tuple $(c_r, \text{sig}_{\text{TSA}}(c_r))$ to the PA. The PA checks if the message is of the right form. If it is, he sends the TSA the second message and publishes

$$\alpha_r = (c_r, \text{sig}_{\text{TSA}}(c_r), \text{sig}_{\text{PA}}(c_r, \text{sig}_{\text{TSA}}(c_r)))$$

on an accessible authenticated medium (e.g., in a newspaper). The TSA checks if the second message is of the right form. If it is, he waits for the publication and then checks if the published value α'_r is equal to α_r . If it is not, he sends α_r and the newspaper with α'_r to the judge.

STAMP COMPLETION PROTOCOL. See Fig. 4(c). At the end of the r th round, A gets the value α_r from the medium, checks if it is of correct form, and sends the request v to the TSA. The TSA returns the second message m . The client checks if the signature is correct. If it is, he computes the cumulative hash $L_{|V|}$ from $\text{Cert}(v)$ and compares it against the value in c_r . If the verification algorithm V rejects on input $\text{Cert}(v)$, the client sends α_r and the message m to the court.

The above of protocols rebalances the communication, compared these in [BLLV98], by moving the burden from the stamping protocol to the stamp completion protocol, with the intuition that a lightweight stamping protocol helps to avoid trivial reordering attacks. Because of the collision-resistance of hash function h , transmission of L_v is sufficient if G is a simply connected authentication graph.

7 Optimality

As informally argued in Section 4, the round graph $G = (V, E)$ of a time-stamping service has to be based on a simply connected graph for which the corresponding ϕ -scheme is such that for any source v , L_1 is included in v 's certificate. More precisely, for any source v , there has to be a node w in every root path of v , such that $(1, w) \in E$ and w is computable from $(1, \dots, v)$ but *not* computable from $(1, \dots, v - 1)$. We say that a graph satisfying these conditions is *good*.

For a good graph G we define $\mathbf{a}(G) = \max_{v \in S(G)} \min_{\mathcal{P}} |\mathbf{A}_G(\mathcal{P}(v))|$, where \mathcal{P} ranges over all pass-through functions. Recall that $h : \{0, 1\}^* \rightarrow \{0, 1\}^k$. Then by the definition of certificates, $|\text{Cert}(v)| = k(\min_{\mathcal{P}} |\mathbf{a}_G(\mathcal{P})| + 1)$, and thus $\max_v |\text{Cert}(v)| = k(\mathbf{a}(G) + 1)$. Hence, a tight lower bound on $\mathbf{a}(G)$ yields a tight lower bound on $\max_v |\text{Cert}(v)|$. If G is a round graph, then the number of time-stamps issued per round is equal to $n = |S(G)| - 1$, since one source corresponds to the previous round stamp. Next we prove that for any good graph G , $\mathbf{a}(G) \geq \log_2 n + 2$. Hence, time-stamping service based on threaded tree schemes is certificate length optimal if we cannot assume anything more but existence of (one) collision-resistant hash-function.

Theorem 1. (1) For a rooted directed acyclic graph G , $\mathbf{a}(G) \geq \log_2 |S(G)| + 1$.
(2) For a good graph G , $\mathbf{a}(G) \geq \log_2(|S(G)| - 1) + 2$.

Proof. (1) We show in several steps how a given rooted directed acyclic graph G can be transformed by local modifications to a complete binary tree T_d such that $|S(T_d)| \geq |S(G)|$ and $\mathbf{a}(T_d) \leq \mathbf{a}(G)$.

First step (eliminating fan-in > 2). Replace any vertex with $s > 2$ incoming edges with a binary tree with s sources (Fig. 5, 1). Let G_1 be the resulting graph. Then clearly $|S(G_1)| = |S(G)|$ and $\mathbf{a}(G_1) \leq \mathbf{a}(G)$.

Second step (eliminating fan-out > 1). Every vertex with fan-out > 1 can be replicated (Fig. 5, 2). An application of this step pushes the vertices with fan-out > 1 “downwards” in the graph. The final applications just replicate the sources, without adding any vertices with fan-out > 1. Thus, we can repeat the procedure until we get a binary tree G_2 with no vertex having fan-out > 1. Trivially, $|S(G_2)| \geq |S(G_1)|$ and $\mathbf{a}(G_2) = \mathbf{a}(G_1)$.

Third step (making G_2 complete). Any vertex $v \in V(G_2)$ with only one predecessor can be deleted (Fig. 5, 3). Let the resulting graph be G_3 . Trivially, $|S(G_3)| = |S(G_2)|$ and $\mathbf{a}(G_3) = \mathbf{a}(G_2)$.

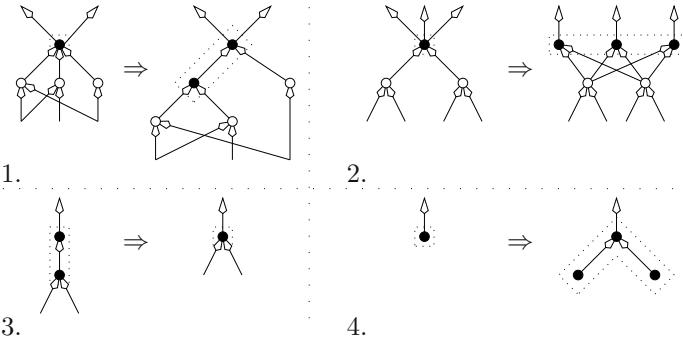


Fig. 5. Local graph modifications in Theorem 1.

Fourth step (balancing G_3). If G_3 is not yet a complete binary tree, there exists a source $v \in V(G_3)$ such that $d(v, |V(G_3)|)$ is less than the height of G_3 . We proceed by adding two vertices as predecessors of v (Fig. 5, 4). Let the resulting graph be G_4 . Trivially, $|\mathcal{S}(G_4)| \geq |\mathcal{S}(G_3)|$ and $\mathbf{a}(G_4) = \mathbf{a}(G_3)$.

Thus for any graph G there exists a complete binary tree T_d such that $|S(T_d)| \geq |S(G)|$ and $\mathbf{a}(T_d) \leq \mathbf{a}(G)$. But $|S(T_d)| = 2^d$ and $\mathbf{a}(T_d) = d + 1$, thus for any graph G , $\mathbf{a}(G) \geq \mathbf{a}(T_d) = \log_2 |S(T_d)| + 1 \geq \log_2 |S(G)| + 1$. \square

(2) Let $G = (V, E)$ be a good graph with $n = |S(G)| - 1$. W.l.o.g. we can assume that the minimal authenticator for any $v \in S(G)$ has the cardinality $\mathbf{a}(G)$. Now let G_1 be the subgraph of G spanned by $V \setminus \{1\}$. By the definition of good graphs, $\mathbf{a}(G_1) = \mathbf{a}(G) - 1$. Since $\mathbf{a}(G_1) \geq \log_2 |S(G_1)| + 1$, we get that $\mathbf{a}(G) \geq \log_2 n + 2$. \square

8 Conclusion

The new time-stamping scheme achieves optimal lengths of time certificates in a general sense ($k \log_2 n$ versus $3.786k \log_2 n$ compared to the optimal anti-monotone scheme [BL98], where n is the number of time stamps issued during a round). As compared to the formerly known most efficient tree schemes, the new scheme has the same space complexity and only somewhat larger time complexity, while being accountable like the scheme of [BLLV98]. We also presented the first accountable publication protocol that makes it unnecessary to audit the publication process.

In practice, a reliable time-stamping service is virtually impossible to achieve if the TSA and the PA are not replicated. Further research in this area is definitely necessary. The formal notion of temporal security is not yet developed and needs additional research; it is not even clear if it could be defined separately from the security of other primitives. An interesting question is whether more efficient time-stamping services could be built on stronger cryptographic assumptions.

Acknowledgements

The authors are thankful to Stuart Haber, Markus Jakobsson, Peeter Laud, Kaisa Nyberg and to the anonymous referees for feedback and helpful discussions.

References

- [BdM91] Josh Benaloh and Michael de Mare. Efficient Broadcast Time-Stamping. Technical Report 1, Clarkson University Department of Mathematics and Computer Science, August 1991. [296](#)
- [BdM93] Josh Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures (Extended Abstract). In Tor Helleseth, editor, *Advances in Cryptology—EUROCRYPT 93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer-Verlag, 1994, 23–27 May 1993. [296](#), [299](#)
- [BHS92] Dave Bayer, Stuart A. Haber, and Wakefield Scott Stornetta. Improving the Efficiency And Reliability of Digital Time-Stamping. In *Sequences '91: Methods in Communication, Security, and Computer Science*, pages 329–334. Springer-Verlag, 1992. [296](#)
- [BL98] Ahto Buldas and Peeter Laud. New Linking Schemes for Digital Time-Stamping. In *The 1st International Conference on Information Security and Cryptology*, pages 3–14, Seoul, Korea, 18–19 December 1998. Korea Institute of Information Security and Cryptology. [294](#), [296](#), [299](#), [301](#), [304](#)
- [BLLV98] Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-Stamping with Binary Linking Schemes. In Hugo Krawczyk, editor, *Advances on Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 486–501, Santa Barbara, USA, August 1998. Springer-Verlag. [294](#), [296](#), [297](#), [298](#), [299](#), [301](#), [302](#), [303](#), [304](#)
- [HS90] Stuart Haber and Wakefield Scott Stornetta. How to Time-Stamp a Digital Document. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 437–455. Springer-Verlag, 1991, 11–15 August 1990. [294](#), [295](#)
- [HS91] Stuart A. Haber and Wakefield Scott Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2):99–111, 1991. [294](#)
- [Mer80] Ralph C. Merkle. Protocols for Public Key Cryptosystems. In IEEE, editor, *Proceedings of the 1980 Symposium on Security and Privacy, April 14–16, 1980 Oakland, California*, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1980. IEEE Computer Society Press. [295](#)
- [San99] Tomas Sander. Efficient Accumulators without Trapdoor. In *The Second International Conference on Information and Communication Security*, Sydney, Australia, 9–11 November 1999. To appear. [296](#)

“Pseudorandom Intermixing”: A Tool for Shared Cryptography

Yair Frankel¹, Philip MacKenzie², and Moti Yung³

¹ CertCo, N.Y., NY, yfrankel@cryptographers.com, yfrankel@cs.columbia.edu

² Bell Laboratories, Murray Hill, NJ, philmac@research.bell-labs.com

³ CertCo, N.Y., NY, moti@certco.com, moti@cs.columbia.edu

Abstract. Designing distributed cryptographic protocols that combine correctness, security, efficiency and practical constraints can be very difficult. Here, we suggest a new modular tool that we call “*pseudorandom intermixing*” which allows parties (or architectural components, such as hardware devices) sharing pseudorandomness to mix extra correlated pseudorandom information inside their computational results. We show how the pseudorandom intermixing may ease the design, increase efficiency and allow more refined control of cryptographic protocols for several important tasks, while maintaining “provable security.” It can even turn a “heuristic protocol” into a “provably secure” one.

We concentrate on the area of “distributed public key systems,” which has been a very active area of research in the last decade, and for which there is a great interest in practical implementations of protocols. Among other things, we demonstrate the first “fault-free non-interactive” proactive maintenance protocol for RSA, which involves a single broadcast round to perform an update, if parties do not behave maliciously. We also demonstrate how to interlace access control within the messaging of proactive RSA; this assures elimination of corrupted entities.

1 Introduction

Cryptographic protocol development is often a challenge for it requires careful design to assure both security and efficiency. Therefore, general design mechanisms which can be incorporated into protocols in a modular fashion and provide easier proofs of security are highly desired.

In this work we develop a tool which we call *pseudorandom intermixing*. In a nutshell, it is a notion that assures *entangling* of pseudorandom information shared between parties with the *cryptographic (algebraic) computational results themselves!* Namely, the pseudorandom intermixings change the local outcomes of computations by adding pseudorandom elements to the computation, in a way much similar to computing with a randomized key schedule.

As a result of pseudorandom intermixing, the local outputs appear random, which makes it easier to simulate the protocol and prove its security. However, since the outputs are not random, but rather t -wise random, and the intermixing is deterministically generated (pseudorandom), they allow for handshaking (assurance of collaboration) amongst all the active players.

A major area of applications of pseudorandom intermixing is distributed cryptosystems, an important field which enables distributed trust, added security, increased availability, and avoidance of single points of failure (for surveys on the subject see [D92,FY98]). In particular we concentrate on distributed systems based on RSA [RSA], which is perhaps the most widely used public-key function. Such distributed RSA systems have been shown possible in various security/adversarial models [B88,F89,DF91,DDFY,FGY,GJKR96,FGMY,FGMY2,R]. The cryptographic objective of a distributed threshold RSA function-sharing system is to distribute the RSA signing capability so that any t or more entities can sign a message, yet an adversary that compromises at most $t - 1$ entities can not sign.¹

1.1 The Basic Issues

In addition to considering enhanced security and efficiency of cryptographic protocols, we also consider enhanced control aspects which allow the parties better tools to assure policy adherence.

In fact, many of the previous threshold cryptosystems missed vital control aspects necessary for a sound and secure deployment when taking into consideration the complete systems perspective and not just the purely mathematical aspects of distributed function application. Here we resolve many of the open issues of a secure system design for threshold cryptosystems by adding control into the operation without adding much cost. In fact, we interlace access control management into the design. To demonstrate the need, let us suppose that an entity has been found to be corrupted (for example, an employee who is quitting or being fired, an entity who lost a key due to a virus attack, etc.). Then, in several threshold cryptosystems, when the adversary has the outputs of $t - 1$ good parties then the adversary can make the signature. But, this is not what the policy should allow! Rather, it should require a t quorum of good components. It should further require that when one is identified as corrupt then that entity should have no capabilities whatsoever. Indeed, proactive system [OY] resolve the problem of cleaning up the system and ignoring components known to be corrupt. However, they require all entities to be present and participate in a costly update protocol in order to effectively remove a party. A better solution would be to require the working parties to perform a handshake in order to know and validate whom they are working with. Though completely non-interactive protocols (e.g., [F89,DDFY,Sho]) may be efficient and mathematically interesting, there are situations where they may have reduced security when corrupted players are identified and when the policy requires to ignore them. (In addition, the system audit mechanisms may require anyway to know which elements are active). Pseudorandom intermixing interlaces t -wise access control to allow the protocol to be non-interactive except for the announcement of the active players

¹ This is the same protection as in secret sharing [Bl,Sh], but the signing operation may be repeated as long as the key is valid.

(entities) within a round or a period and yet to enable access control based on known dishonest/unavailable parties.

There are various other operational aspects. One important aspect of systems is that the system must have minimal availability and security requirements yet require minimal resources. Let us give an example based on the proactive RSA ramp² scheme [R]. This system in its simple form allows for a reduction in the available players to perform future signatures when an entity becomes inactive yet uncorrupted. Moreover, to turn the RSA ramp scheme back into an RSA threshold scheme, we require (for a scheme with a total of l entities) either private memory of size $O(l^3)$ or interaction of all players after each signature via a proactive update, making it into an l -out-of- l system (visualize the availability requirement change when $l=100$ and $t=3$). Both measures result in extremely cumbersome systems from any practical perspective and the simplicity of the simplified approach is lost. Another example is the non-interactive robustness assurance of [GJKR96] with check vectors. This method requires incorporating additional security requirements on parties which normally are treated as unsafe (e.g., the combiner is required to have secret keys—this is against the original model where the combiner is merely an abstraction). Such a single point of failure is dangerous—it requires a unique combiner to be available to assure robustness of operation.

The above examples, show that many designs which are legitimate mathematical exercises resulting in new ideas which may even optimize a single parameter (and we do not object to such investigations per se), may nevertheless be undesirable in actual systems. The balance of availability, security, efficiency and flexible control is essential in the design of practical systems.

1.2 Our Results

Our specific results on distributed cryptosystems include:

Security By incorporating pseudorandom intermixing, we are able to convert the simple yet heuristic threshold RSA protocol of [DF91] into a provably secure protocol. Pseudorandom intermixing is able to achieve this by converting public values which are not known to be simulatable into random-looking values that are easy to simulate.

Efficiency By incorporating pseudorandom intermixing, we are able to reduce the communication requirements of the secure threshold RSA protocol of [FGMY2] and reduce the communication of robustness/checking information in the robust threshold RSA of [FGMY2]. Moreover, we show how the shared pseudorandomness enables elimination of interaction, turning interactive protocols into “fault-free non-interactive” ones (a notion of efficient protocols put forth in [F]). In particular, we show how to use pseudorandom intermixing to eliminate interaction in the distributed RSA signing protocol of [FGMY2] and give the first non-interactive proactive update.

² See [BM] for a reference to ramp schemes; essentially it allows for reduced threshold throughout the system’s operation.

Access Control Pseudorandom intermixing allows easy access control management, where parties can be included (excluded) from a designated “working set” by including (excluding) the parties’ shared pseudorandomness from the computation. As a result, the correctness of the final result in a distributed computation implies the correct working set is involved.

Isolation Pseudorandom intermixing individualizes each computation, by effectively changing the key based on the common message. This can help in foiling side-channel cryptanalysis (based on timing or power consumption leakage).

We feel that the practice of cryptographic systems needs generic tools that can, in a modular fashion, enhance various properties of numerous protocols and working environments (as we mention above). Pseudorandom intermixing is such a tool and is based on mixing public and private key cryptography in a new way. It seems to us that perhaps it may provoke more thinking about new cryptographic tools with wide modular applicability.

We comment that the use of shared randomness in distributed protocols and how it enables certain designs was recently presented (in an independent work) [GI99] where shared randomness is treated as a “compressed resource.” They used the resources for general multi-party computation and proactive secret sharing. Our work has a bit different flavor: we show how the resource can be integrated into “existing protocols” in a modular fashion, thus showing the “incremental nature” of the resource, also our area of application is the more pragmatic “distributed cryptosystems.” In addition, we pseudorandomly derive the correlated randomness from common inputs, and thus effectively “individualize” each computation, which may help against side-channel attacks (e.g. timing attacks). The general idea of building basic tasks from shared randomness was first put forth in a generic theoretical context in [B97].

Organization: We discuss some basic tools we use in Section 2, and we provide an informal description of pseudorandom intermixing in Section 3. We then present numerous applications of pseudorandom intermixing to threshold cryptography in Section 4. Finally, in Section 5 we give applications to fault tolerant protocols (robust and proactive ones).

2 Background

Here we describe the basic cryptographic functions and components that we use in the paper.

The RSA algorithm: The RSA key generator produces two large random primes p_1 and p_2 , and computes a composite $n = p_1 \cdot p_2$ and the Euler totient function $\phi(n) = (p_1 - 1)(p_2 - 1)$. The generator chooses a public key $\langle e, n \rangle$, where $\gcd(e, \phi(n)) = 1$, and private key d , such that $ed \equiv 1 \pmod{\phi(n)}$. The one-way (hard) direction of RSA (used in decryption or signature of a message m) is $S_m \equiv m^d \pmod{n}$, whereas the public (easy) direction (used in encryptions and verification of signature) is the inverse function $z^e \pmod{n}$ for a message

z. Typically for signatures, the message m to be raised to the power d is a cryptographic hash of the real message M .

Discrete logarithm: Let P be a prime and g a generator of a subgroup of Z_P of large order. The function $f(x) = g^x \bmod P$ is one-way (finding x is called the discrete log problem). The Diffie-Hellman key exchange [DH] can be built on it. We can similarly define discrete logarithms over composites.

Pseudorandom functions: With a pseudorandom function family, given a random member of the family the result at any chosen input point looks random, and sampling it polynomially many times cannot distinguish it from a truly random function. We often denote a pseudorandom function family as PRF and an element from PRF indexed by k as PRF_k . Formal definitions are in [GGM]. Recently, designing pseudorandom functions from pseudorandom permutations (block ciphers) has been discussed in [HWKS].

Distributed Cryptography: We deal with systems where the capability to apply a cryptographic function (an RSA private function, in our case) is distributed in a threshold scheme. We have a group of l servers who securely share a private key d via shares s_1, \dots, s_l . The servers are connected to a common broadcast medium C , called the communication channel, with the property that messages sent on C reach every party connected to it. We assume that each server has a local source of randomness and that the system is synchronized (and w.l.o.g. that servers act synchronously). When a quorum of t shareholders are available they can cooperate to compute the function, yet less than a quorum cannot break the system. Formally, in our security proofs we assume the corrupting adversary is non-adaptive throughout the lifetime of the system and is restricted to corrupt at most $t - 1$ shareholders. Such an adversary is called a *$(t - 1)$ -restricted adversary*.

In a **threshold RSA system**, the function computation consists of a quorum of t servers who obtain an input m , generate partial signatures for m , and output these results to a *not* necessarily trusted combiner (which may simply be a designated server). The combiner, using the quorum of t partial signatures and input m generates the signature of the message, $m^d \bmod n$.³ After signing a message, security is maintained in the sense that a quorum of t servers is again needed to sign a new message. When the signing operation is guaranteed (with high probability) to be polynomial-time under arbitrary malicious behavior of a $(t - 1)$ -restricted adversary, the system is called **robust**. If the system is maintained against a mobile adversary it is called **proactive**.

3 Pseudorandom Intermixing

The idea behind pseudorandom intermixing is simple. It involves entangling together (1) a cryptographic computation, and (2) a computation on pseudorandom numbers, such that the combined result is the same as that of the cryp-

³ We note that since we do not put trust in the combiner, (1) it must be the case that the partial signatures do not provide any information that could help an adversary sign a new message, and (2) the combiner certainly should not hold any private keys.

tographic computation alone. In this paper, the cryptographic computation will generally be the computation of certain partial results of a distributed computation that will be summed together, and we will intermix (add in) pseudorandom numbers that sum to zero. Thus the result of the cryptographic computation will be unaffected by the pseudorandom numbers, although the “partial results” will be, in effect, randomized.

Here we give an example. Assume there is a set of users indexed by a set Λ , and each user i has a share s_i of a secret s where $s = \sum_{i \in \Lambda} s_i$. Also assume that each pair of users (i, j) shares a value $p_{i,j} = p_{j,i}$, which for now we may assume was chosen randomly from a large domain. It is easy to see that

$$\sum_{i \in \Lambda} \sum_{j \in \Lambda, j \neq i} p_{i,j} \cdot \text{sign}(j - i) = 0,$$

since each $p_{i,j}$ cancels $p_{j,i}$ (because they are added together with opposite signs). Now each user i can compute a pseudorandom intermixing value $r_i = \sum_{j \in \Lambda, j \neq i} p_{i,j} \cdot \text{sign}(j - i)$, and it follows that $\sum_{i \in \Lambda} r_i = 0$. User i may now compute an entangled output $s_i + r_i$, and we note that the sum of the entangled outputs is the secret:

$$\sum_{i \in \Lambda} (s_i + r_i) = \sum_{i \in \Lambda} s_i + \sum_{i \in \Lambda} r_i = s + 0 = s.$$

The final result is the same as the non-entangled computation, so why complicate the process? The issue is that during the process, the individual entangled outputs “look” random, and as we will show in the next section, this enables easier proofs of security in some involved distributed protocols.

We note that the $p_{i,j}$ values will actually be generated by a pseudorandom function. That is, each pair of users (i, j) will share a key $\sigma_{i,j} = \sigma_{j,i}$ to a pseudorandom function PRF, and they generate $p_{i,j} = \text{PRF}_{\sigma_{i,j}}(m)$ where m is a tag for the intermixing, such as a message to be signed. This tag serves a dual purpose of verifying that all users are working on the same computation at the same time.

The shared keys to PRF are relatively easy to generate using Diffie-Hellman key exchange. They can also be considered “disposable,” since revealing them does not compromise the security of the main cryptographic function. We will see an example of this when we discuss the robust threshold RSA protocol.

Efficiency of size of keying information: It should be noted that even though extra keys are needed for pseudorandom intermixing with distributed RSA, our system is still storage efficient for practical implementations. In particular, we will deal with distributed cryptosystems where each server has a constant number of RSA key shares and thus a total of $O(l)$ RSA key shares must be stored. There are $O(l^2)$ pseudorandom intermixing keys shared in the entire system, but these are typically symmetric cryptography keys which are perhaps an order of magnitude smaller than the RSA key shares. (Recall that in [R] the threshold scheme requires $O(l^3)$ RSA keys in order to operate in a fault-tolerant threshold scheme without degradation of security or availability.)

We next investigate the application of pseudorandom intermixing to threshold RSA protocols. We remark that the method also applies to Discrete Log based systems in an analogous way. In Section 4.1 we convert the heuristic threshold RSA protocol from [DF91] into a provably secure protocol. In Section 4.2 we improve the efficiency of the secure threshold RSA protocol from [FGMY2]. In Section 5.1 we improve the efficiency of the robust threshold RSA protocol from [FGMY2]. A non-interactive proactive protocol in the fault free case is given in Section 5.2.

4 Applications to Threshold Cryptography

4.1 Pseudorandom Intermixing as Design Tool: Assuring Provable Security

In Crypto '91, [DF91] developed a heuristically secure threshold RSA scheme (see Appendix) in which servers do not need to communicate with each other to sign a message (i.e., it is a non-interactive threshold signing where outputs of servers which only have to know the working set, go to the combiner). The system is not known to be secure because it has never been proven whether the partial results sent from signing servers to the combiner S do not constitute sufficient information to allow for an adversary (which may hold S and up to $t - 1$ servers) to sign new messages. Here we demonstrate how pseudorandom intermixing can take a protocol which is not known to be secure and make it into a secure one.

The protocol in [DF91] works in the following way. Server i obtains share s_i from a trusted key distributor. The scheme is heuristically secure since it is not known how an adversary with possession of $t - 1$ or less servers can learn useful information that will compromise the system. Now, to sign a message m with group $\Lambda = \{i_1, \dots, i_t\}$, each server i first generates $s''_{i,\Lambda}$ from s_i and the current active group Λ , (it holds that $d - 1 = \sum_{j \in \Lambda} s''_{j,\Lambda}$). Server i outputs $S_{m,i,\Lambda} = m^{s''_{i,\Lambda}} \bmod n$ to the Combiner. The Combiner can now compute signature $m^d \equiv m \prod_{j \in \Lambda} S_{m,j,\Lambda} \bmod n$. This latest process is not known to be simulatable.

We now modify [DF91] using pseudorandom intermixing .

1. Each pair of servers (i, j) share a private key $\sigma_{i,j} = \sigma_{j,i}$ for a pseudorandom function.
2. When applying a function on input m :
The partial result now becomes $S_{m,i,\Lambda} = m^{s'_{m,i,\Lambda}} \bmod n$ where $s'_{m,j,\Lambda} = s''_{j,\Lambda} + [\sum_{v \in \Lambda \setminus \{j\}} \text{sign}(j - v) \cdot \text{PRF}_{\sigma_{j,v}}(m)]$.
3. The combiner computes $m \prod_{j \in \Lambda} m^{s'_{m,j,\Lambda}} \equiv m^d \bmod n$ (since $\sum_{j \in \Lambda} s'_{m,j,\Lambda} = d - 1$).

Fig. 1. Provably Secure DF91+pseudorandom intermixing

This slightly and modularly modified protocol (where fields in communicated messages did not change) is now secure:

Theorem 1. *Breaking the protocol of this subsection by a $(t - 1)$ -restricted adversary implies breaking RSA or that PRF is not a pseudorandom function family (i.e., the protocol is a secure threshold RSA protocol).*

Proof. (Sketch) We prove that if an adversary having possession of up to $t - 1$ of the servers and the combiner can break the protocol then it can break RSA. We use a standard argument that the system can be simulated with input given to an attacker on the direct (non-distributed RSA scheme) who is allowed to probe the system on messages m_1, \dots, m_s and gets the signatures. The simulator then gets $\langle n, e, (m_1, m_1^d), \dots, (m_s, m_s^d) \rangle$ where $s = \text{poly}(h)$ for security parameter $h = \log(n)$. Let $\Lambda_i = \{j_{i,1}, \dots, j_{i,t}\}$ be the set of servers working together to sign message m_i . Moreover, let the adversary control the combining functions as well as servers indexed by A with shares of d such that $|A| \leq t - 1$. We note that the distribution of the adversary receiving up to $t - 1$ shares is simulatable.

When $|\Lambda_i \cap A| = t - 1$ then trivially the simulator can compute $m_i^d / (m_i * \prod_{i \in A} m_i^{s'_{m_i, i, \Lambda_i}}) \bmod n$, simulating the output of the server the adversary does not control. In fact, also [DF91] can be simulated in this case. Now we discuss the case in which it is not known how to simulate in [DF91]. When $|\Lambda_i \cap A| \neq t - 1$ then the output of any $i \notin A$ is pseudorandom due to the definition of S_{m_i, i, Λ_i} (except that the sum of all t of them adds up to the given $m^d / m \bmod n$). Hence, we can simulate by choosing random elements $R_1, \dots, R_{t-1-|A|} \in_R Z_n^*$ as the outputs for $t - 1 - |A|$ of the servers that are not controlled by the adversary, and choosing the last server's output as $m_i^d / ((m_i) \cdot (\prod_{i \in A} m_i^{s'_{m_i, i, \Lambda_i}}) \cdot (\prod_{j=1..t-1-|A|} R_j)) \bmod n$.

Now if the adversary, given the simulation, can break the system, then either: the simulation is indistinguishable from a real attack (the functions are pseudorandom), and breaking the distributed system implies breaking RSA. Otherwise, if the probability of breaking during the simulation is different from the probability of breaking in the actual attack differs by more than a negligible amount, this can be turned into a distinguisher for the “assumed pseudorandom function” which contradicts its pseudorandomness.

We note that if corruption is monotonic (rather than static where all corruption are at the start) we have to restart the simulation after each corruption. The above assumed the simulation stage after all corruptions are known.

Another nice property of the above modification is that there is no change in the interaction or messages used in the protocol. This is very important in cases where an old heuristic protocol has been implemented and the new secure one can be easily retrofitted .

4.2 Intermixing Assures Efficiency of Secure Threshold RSA

Next we describe the secure threshold protocol in [FGMY2]. This protocol has a setup phase (as in Figure 2, but without Step 3) where each server j obtains share s_j from a trusted dealer or via a distributed key generation protocol.

Now, to sign a message m in the secure threshold RSA protocol with group $\Lambda = \{i_1, \dots, i_t\}$, each server j first generates $s_j z_{j,\Lambda}$ (where $z_{j,\Lambda}$ is a constant used for polynomial interpolation and is computable given j and Λ). Now it holds that $d = P + \sum_{j \in \Lambda} (s_j z_{j,\Lambda} + R_{j,m,\Lambda})$ where $R_{j,m,\Lambda}$ is a random string jointly generated (as discussed below) by the servers in Λ for server j . Server j outputs $S_{m,j,\Lambda} = m^{s_j z_{j,\Lambda} + R_{j,m,\Lambda}}$. The Combiner can now compute signature $m^d \equiv m^P \prod_{j \in \Lambda} S_{m,j,\Lambda} \pmod{n}$.

The generation of the $R_{j,m,\Lambda}$ values is a communication intensive protocol. Each party $j \in \Lambda$ first commits to $|\Lambda|$ random values via an information theoretically secure commitment scheme. This requires at least $2|\Lambda|$ exponentiations. Even more, a zero-knowledge proof is used to demonstrate that each commitment was performed correctly. Finally, there is a private transmission of the random values from each shareholder to the other shareholders in Λ . The idea is to change the polynomial sharing into a additive t -out-of- t sharing inside the group (changing the secret representation method is crucial in the work). Fortunately, the $R_{j,m,\Lambda}$ values can be cached, and thus they only need to be computed once for each Λ . However, this may require the shareholder to store significant amount of sensitive data.

Setup: The following one time setup is performed.

1. The (centralized/distributed) dealer generates an RSA with a public key (e, n) and a private key d .
2. Using the extended Euclidean algorithm, the dealer computes P, s' such $1 = eP + \frac{L^2}{H^2}s'$ where $L = l!$ and $H = \gcd(e, L)$. Note that $d \equiv P + L^2 \cdot k' \pmod{\phi(n)}$ where $k' \equiv ds'H^{-2} \pmod{\phi(n)}$. The dealer chooses a random polynomial $A(x) = A_0 + A_1x + \dots + A_{t-1}x^{t-1}$, such that $A(0) = A_0 = L^2 \cdot k'$ and $A_j \in_R \{0, L, \dots, 2L^3n^{2+\epsilon}t\}$ for $1 \leq j \leq t-1$. (All operations are performed over the integers.) All servers receive public point P , and each server i with public interpolation point $x_i = i$ receives secretly shadow $s_i = A(x_i) \in Z$.
3. (**Pseudorandom intermixing setup**) Each pair of servers (i, j) obtains a shared secret intermixing key $\sigma_{i,j} = \sigma_{j,i}$ for the pseudorandom generator. (In a distributed key generation protocol this can be performed via Diffie-Hellman key exchange [DH] with added authentication or some other shared randomness generation technique, but for now one may simply assume the keys are generated by the dealer.)

Notation: $z_{j,\Lambda} = \prod_{v \in \Lambda \setminus \{j\}} (x_j - x_v)^{-1} (0 - x_v)$

Fig. 2. Setup for a secure threshold RSA protocol with pseudorandom intermixing

Using pseudorandom intermixing, we may replace this interactive process of “generating randomness” with a non-interactive process. Moreover, the new process will require significantly less local computation.

For the new process we need a new setup protocol, shown in Figure 2. Observe that step 3 is the setup for the pseudorandom intermixing; the rest is the same as in [FGMY2].⁴ The new signing protocol is shown in Figure 3. Observe that step 1 uses pseudorandom intermixing to create the “random-looking values” that were previously created through an interactive protocol.

Signing (operational) Phase: Let Λ where $|\Lambda| = t$ be the servers designated to participate in the signing. The Combiner sends a description of the set Λ to all members of Λ .

1. Each server j computes $s'_{m,j,\Lambda} = s_j \cdot z_{j,\Lambda} + \{ \sum_{v \in \Lambda \setminus \{j\}} \text{sign}(j - v) \cdot \text{PRF}_{\sigma_{j,v}}(m) \}$
2. Each server j transmits partial signature $S_{m,j,\Lambda} \equiv m^{s_{m,j,\Lambda}} \pmod{n}$ and transmits the result to the Combiner.
3. The Combiner computes the signature for m from the partial signatures, $S_m \equiv P \cdot \prod_{v \in \Lambda} S_{m,v,\Lambda} \pmod{n}$.
4. **(Validate implicit hand-shake)** The Combiner verifies: $(S_m)^e \stackrel{?}{\equiv} m \pmod{n}$.

Fig. 3. Secure non-interactive threshold RSA signature generation with pseudorandom intermixing

Theorem 2. *The non-interactive randomized signing protocol in this subsection produces a correct RSA signature corresponding to public key (e, n) .*

Proof. Note for any j where $1 \leq j \leq l$, L is divisible by $l \cdot \prod_{i \in \{1, \dots, j-1, j+1, \dots, l\}} (x_j - x_i)$. Let $H = \gcd(e, L)$ and observe that H^{-1} exists modulo $\phi(n)$ because e^{-1} exists. Let $1 = eP + \frac{L^2}{H^2}s'$. Note $d \equiv P + L^2 \cdot k' \pmod{\phi(n)}$ where $k' \equiv ds'H^{-2} \pmod{\phi(n)}$. Hence, over the rationals $d - P = L^2k' = \sum_{i \in \Lambda} s_i \cdot z_{i,\Lambda}$ by the property of Lagrange interpolation [Sh]. Moreover, it is computable over the integers since $s_i \cdot z_{i,\Lambda}$ is an integer because L divides s_i . Hence,

$$\begin{aligned} d - P &= \sum_{i \in \Lambda} s_i \cdot z_{i,\Lambda} \\ &= \sum_{i \in \Lambda} s_i \cdot z_{i,\Lambda} + \sum_{i \in \Lambda} \left(\sum_{v \in \Lambda \setminus \{i\}} \text{sign}(i - v) \cdot \text{PRF}_{\sigma_{i,v}}(m) \right) \end{aligned}$$

⁴ Throughout our discussions we assume a trusted dealer in order to simplify our discussion. However, we should note that using [BF97, FMY] one can employ a distributed dealer procedure among the shareholders to initiate the current protocol, hence not relying on any single entity (dealer).

$$\begin{aligned}
&= \sum_{i \in \Lambda} \left(s_i \cdot z_{i,\Lambda} + \sum_{v \in \Lambda \setminus \{i\}} \text{sign}(i - v) \cdot \text{PRF}_{\sigma_{i,v}}(m) \right) \\
&= \sum_{i \in \Lambda} s'_{i,m,\Lambda},
\end{aligned}$$

since $(\text{sign}(i - v) \cdot \text{PRF}_{\sigma_{i,v}}(m)) + (\text{sign}(v - i) \cdot \text{PRF}_{\sigma_{v,i}}(m)) = 0$. Therefore, $m^d \equiv m^{P+L^2k'} \equiv m^P \cdot \prod_{i \in \Lambda} m^{s'_{i,m,\Lambda}} \pmod{n}$.

We now discuss the security of the protocol described in Figures 2 and 3 in the stationary adversary model, in which an adversary corrupts servers statically (we may also allow corruption in a non-adaptive monotonic fashion and restart the simulation from scratch after each corruption (monotonic means that once the adversary corrupts a server the server remains corrupted throughout). Moreover, we assume the adversary is $(t - 1)$ -restricted. The proof is similar to that of the previous section.

Theorem 3. *The protocol of this subsection is a secure threshold RSA protocol, i.e., if a $(t - 1)$ -restricted adversary can break the protocol, then RSA can be broken or PRF is not a pseudorandom function family.*

Proof. (Sketch) First we prove security assuming each $\text{PRF}_{\sigma_{i,j}}$ is a truly random function. We use a standard argument that the system can be simulated with input $\langle n, e, (m_1, m_1^d), \dots, (m_s, m_s^d) \rangle$ where $s = \text{poly}(h)$ with security parameter $h = \log(n)$. Let $\Lambda_i = \{j_{i,1}, \dots, j_{i,t}\}$ be the set of servers working together to sign message m_i . Say the adversary controls the Combiner as well as servers indexed by A where $|A| \leq t - 1$. The simulability of the up to $t - 1$ shares the adversary sees was shown in [FGMY2]. Moreover, it is trivial to simulate the distribution of the (random) keys $\{\sigma_{i,j}\}$. When $|\Lambda_i \cap A| = t - 1$ then the simulator can trivially compute the output $m_i^d / ((m_i^P) \cdot (\prod_{j \in A} m_i^{s'_{m_i,j,\Lambda_i}})) \pmod{n}$ of the server the adversary doesn't control. When $|\Lambda_i \cap A| < t - 1$ then the output of Server j ($j \notin A$) is random by the definition of S_{m_i,j,Λ_i} (except that the sum of all t of them adds up to $m^{d-P} \pmod{n}$). Hence, we can simulate by choosing random elements $R'_1, \dots, R'_{t-1-|A|} \in_R Z_n^*$ as the outputs for $t - 1 - |A|$ of the servers that are not controlled by the adversary, and $m_i^d / ((m_i^P) \cdot (\prod_{i \in A} m_i^{s'_{m_i,i,\Lambda_i}}) \cdot (\prod_{j=1..t-1-|A|} R'_j)) \pmod{n}$ as the output of the last server not controlled by the adversary. Now if one can break this system then they can use the simulation to break RSA using well established proof techniques.

Thus we may assume that the probability of breaking the system, assuming each $\text{PRF}_{\sigma_{i,j}}$ is a truly random function, is negligible. Then it is easy to conclude that if one can break the system, then PRF is not a pseudorandom function family, since the simulation can be made into a polynomial time distinguisher between a function from PRF and a truly random function.

Additional Properties It is often desirable to have additional security properties incorporated into the design of cryptographic protocols. We briefly list

here some additional properties that pseudorandom intermixing provides in the protocol above.

Handshake The intermixing of the results from the pseudorandom functions into the signature computations generates a “ t -wise hand-shake.” This creates a self-awareness property where the absence of a server is detected as long as a single original server is present— it is particularly useful when the servers want to be sure who is participating in the computation. (The protocol in [FGMY2] does provide the handshake but only when the interactive protocol is performed on a per message basis.) Thus the handshake effectively provides “access control” for a distributed computation, where the correctness of the final result implies the correct “working set” performed the computation.

Isolation The pseudorandom intermixing used in generating the partial results provides some protection against timing, power and other side channel attacks [K,KJJ]. Recently in [DKLMQW], an actual implementation of the timing attack was performed and publicly reported. Due to the pseudorandomness in our system, an adversary which looks at the timing (or other side information) of the partial result computation and tries to use the timing channel to deduce the permanent share will fail, since exponentiation is performed using a pseudorandom exponent which differs for each message. (This essentially creates a “random key schedule” for each message.)

5 Application to Fault Tolerant Protocols

Next we investigate applications to robust and proactive protocols.

5.1 Efficiency of Robust Threshold RSA

In the secure threshold RSA protocol above, if a server misbehaves, it may be difficult for the combiner to ascertain *which* server has misbehaved. Trying all subsets of servers until a correct signature is computed may be very expensive (e.g., iteratively attempting to find a subset of $t = l/2 + 1$ honest servers out of l would be exponential in l). Thus we need to make the protocol “robust,” meaning that the protocol should allow the correct signature to be computed efficiently even when up to $t - 1$ parties misbehave. This generally involves adding a method to verify the partial computations of the servers.

Robust RSA protocols were first introduced in [FGY] and then [GJKR96]. However, we will continue to work with the protocol from [FGMY2], but using pseudorandom intermixing to improve the efficiency.

As in [FGMY2], robustness can be added to the protocol from Section 4.2 by including verification information for the partial results in Step 2 in Figure 3. However, we use pseudorandom intermixing to decrease communication by about half compared to [FGMY2]. The protocol is shown in Figure 4. The idea here is to commit to the pseudorandom intermixing keys. When there is an active dispute

(but not when a server is simply unavailable) the commitment is opened to resolve the dispute. Note that it is the key for the pseudorandom function that is opened, and not an RSA key share. This is precisely what we meant in Section 3 by these shared keys being “disposable.” Revealing them does not reveal any RSA shares, and thus does not reduce the security of the RSA signature function. Note also that we do not degrade the size of the quorum needed to sign due to unavailability of servers.

- (Recall System setup) The (centralized/distributed) dealer chooses generators g, g_1 from Z_n^* of maximal order, and where the discrete log of g_1 base g is unknown. Then for each i the dealer publishes $g^{s_i \cdot L^2} \bmod n$. Finally intermixing keys $\sigma_{i,j}$ and $\sigma'_{i,j}$ are generated for $1 \leq i, j \leq l$.
- (Server i setup) Each Server i publishes commitments $g^{\sigma_{i,j}} g_1^r, g^{\sigma'_{i,j}} g_1^{r'}$ to its intermixing keys.
- For a failed attempt in signing a message m by Λ :
 - Server i publishes $S'_{m,i,\Lambda} \equiv g^{L^2 s'_{m,i,\Lambda}} \bmod n$ and $U_{m,i,\Lambda} \equiv \prod_{j \in \Lambda \setminus \{i\}} (g_1)^{\text{PRF}_{\sigma'_{i,j}}(m) \cdot \pi'(i,j)}$ where $\pi'(i,j) = -\text{sign}(i-j)$.
 - For $i \neq j$, Server i or j publishes $R_{m,i,j} \equiv g^{\text{PRF}_{\sigma_{i,j}}(m) \cdot L^2} (g_1)^{\text{PRF}_{\sigma'_{i,j}}(m)}$. (In each pair, only one of Server i or Server j needs to publish, and the other needs to verify.)
 - Dispute resolution: if there is a dispute between Server i and Server j , then they open up their commitments to $\sigma_{i,j}$ and $\sigma'_{i,j}$, and a correct one is used to complete the protocol.
 - Each server j verifies that for all $i \in \Lambda \setminus \{j\}$:

$$(S_i)^{V_{i,1}} \stackrel{?}{=} (S'_{m,i,\Lambda} (U_{m,i,\Lambda})^{-1} \prod_{v \in \Lambda} (R_{m,i,v})^{\pi'(i,v)})^{V_{i,2}}$$
 where $V_{i,1} = \prod_{v \in \Lambda \setminus \{i\}} (0 - x_v)$ and $V_{i,2} = \prod_{v \in \Lambda \setminus \{i\}} (x_i - x_v)$.
 - If the verification does not pass then Server i is removed and a new Λ is chosen to perform the signature.
 - Each Server i performs proofs of knowledge with all other servers of the discrete logs of $S'_{m,i,\Lambda}$ base g^{L^2} and $U_{m,i,\Lambda}$ base g_1 (This may be a non-interactive version of a Fiat-Shamir/Schnorr style proof [FS, Sch], with security based on the “ideal hash” assumption [BR93]).
 - Each Server i proves his partial signature is correct using the robustness algorithm of [FGY] or [GJKR96] (for safe primes), with witness $S'_{m,i,\Lambda}$.
 - If a server is unable to perform any of the ZK proofs, or if it has simply stopped, it is removed and a new Λ is chosen to perform the signature.

Fig. 4. Verification of partial signatures in a secure threshold RSA protocol with pseudorandom intermixing

The protocol in Figure 4 is performed for locating misbehaving servers if signing with the sum-shares produced an invalid signature.

Theorem 4. *The protocol as described in this subsection is a robust threshold RSA protocol, i.e., if a $(t - 1)$ -restricted adversary can break the protocol or prevent a valid message from being signed, then RSA can be broken or PRF is not a pseudorandom function family.*

Proof. (Sketch)

Security We start by assuming that each $\text{PRF}_{\sigma_{i,j}}$ is a truly random function. As in the proof of Theorem 3, we use a standard argument that the system can be simulated with input $\langle n, e, (m_1, m_1^d), \dots, (m_s, m_s^d) \rangle$ where $s = \text{poly}(h)$ with security parameter $h = \log(n)$. (We also assume all corruption are known due to rewinding of the simulation to start). Let $\Lambda_i = \{j_{i,1}, \dots, j_{i,t}\}$ be the set of servers working together to sign message m_i . Say the adversary controls the Combiner as well as servers indexed by A where $|A| \leq t - 1$. The simulation of shares, intermixing keys, and partial signatures is done just as in the proof of Theorem 3. The simulation of the S_j values is done as in [FGMY2], and actually it is shown that the simulator knows $S_j^{1/L}$.

When $|\Lambda_i \cap A| = t - 1$, the simulator can trivially simulate the S'_{m,v,Λ_i} value of the server v the adversary doesn't control by computing $g^d / ((g^P) \cdot (\prod_{j \in A} g^{L^2 s'_{m_i,j,\Lambda_i}})) \bmod n$. When $|\Lambda_i \cap A| < t - 1$ then the S'_{m,j,Λ_i} values for $j \notin A$ are random (except that the product of all t of them is $g^{d-P} \bmod n$). Hence, we can simulate by choosing random elements $R'_1, \dots, R'_{t-1-|A|} \in_R Z_n^*$ as the S'_{m,j,Λ_i} values for $t - 1 - |A|$ of the servers that are not controlled by the adversary, and $g^d / ((g^P) \cdot (\prod_{j \in A} g^{s'_{m_i,j,\Lambda_i}}) \cdot (\prod_{j=1..t-1-|A|} R'_j)) \bmod n$ as the S'_{m,v,Λ_i} value of the last server not controlled by the adversary.

To simulate the $R_{m,j,v}$ and U_{m,j,Λ_i} values, we simply choose the $R_{m,j,v}$ values randomly, and compute the U_{m,j,Λ_i} values that fit the verification equations. We can do this using

$$U_{m,j,\Lambda_i} \equiv \left((S_j^{1/L})^{LV_{i,1}/V_{i,2}} \right)^{-1} S'_{m,j,\Lambda_i} \prod_{v \in \Lambda_i \setminus \{j\}} (R_{m,j,v})^{\pi'(j,v)} \bmod n,$$

since $V_{i,2}$ divides L . We also simulate the zero-knowledge proofs using their respective simulators.

Now if one can break this system then they can use the simulation to break RSA using well established proof techniques.

Thus we may assume that the probability of breaking the system, assuming each $\text{PRF}_{\sigma_{i,j}}$ is a truly random function, is negligible. Then it is easy to conclude that if one can break the system, then PRF is not a pseudorandom function family, since the simulation can be made into a polynomial time distinguisher between the a function from PRF and a truly random function.

Robustness We start by assuming that each $\text{PRF}_{\sigma_{i,j}}$ is a truly random function. Then we have to show that if an adversary causes the signing procedure to fail, we can break RSA. Given an RSA instance, we run the simulator from the

security proof above, except that we use the extractors for the ZK proofs of knowledge.

For signing of a message m to completely fail, it must be that the adversary is able to make sure the signature protocol on m fails without any server being detected as corrupted. (If a server is detected as corrupted, it will be replaced and the signature protocol will continue on a different subset of servers.) By the last ZK proof in the protocol, the exponent on the partial signature $m^{s'_{m,j,A}}$ of server j , matches the exponent of $S'_{m,j,A}$ base g^{L^2} . Also, for the signature to be incorrect, it must be that $\sum_{j \in A} s'_{m,j,A} \neq d - P$. Recall that we have extracted the exponents on $S'_{m,j,A}$ and $U_{m,j,A}$ for all $j \in A \cap A$ from the proofs of knowledge. (For convenience, call them s'_j and u_j .) We also know the exponents on each S_j for $j \in A \cap A$, (since those are the simulated shares), and we know the values of $r_{i,j} = \text{PRF}_{\sigma_{i,j}}(m)$ and $r'_{i,j} = \text{PRF}_{\sigma'_{i,j}}(m)$ for $i \in A \setminus A$ and $j \in A \cap A$, since server i for $i \in A \setminus A$ knows $\sigma_{i,j}$ and $\sigma'_{i,j}$. Then we have

$$\begin{aligned} \prod_{i \in A \cap A} S_i^{V_{i,1}} &\equiv \prod_{i \in A \cap A} (g^{L^2 s_i})^{V_{i,1}} \\ &\equiv \prod_{i \in A \cap A} \left(S'_{m,i,A} U_{m,i,A}^{-1} \prod_{v \in A \setminus \{i\}} (R_{m,i,v})^{\pi'_{i,v}} \right)^{V_{i,2}} \\ &\equiv \prod_{i \in A \cap A} \left(g^{L^2 s'_i} (g_1^{u_i})^{-1} \prod_{v \in A \setminus A} (g^{r_{i,v}} g_1^{r'_{i,v}})^{\pi'_{i,v}} \prod_{v \in A \cap A \setminus \{i\}} (R_{i,v})^{\pi'_{i,v}} \right)^{V_{i,2}} \\ &\equiv \prod_{i \in A \cap A} \left(g^{L^2 s'_i} (g_1^{u_i})^{-1} \prod_{v \in A \setminus A} (g^{r_{i,v}} g_1^{r'_{i,v}})^{\pi'_{i,v}} \right)^{V_{i,2}} \mod n, \end{aligned}$$

and thus

$$\begin{aligned} g^{L^2 \sum_{i \in A \cap A} (V_{i,1}/V_{i,2}) s_i - L^2 \sum_{i \in A \cap A} s'_i - \sum_{i \in A \cap A} \sum_{v \in A \setminus A} r_{i,v} \pi'_{i,v}} \\ = g_1^{-\sum_{i \in A \cap A} u_i + \sum_{i \in A \cap A} \sum_{v \in A \setminus A} r'_{i,v} \pi'_{i,v}} \mod n. \end{aligned}$$

However, if $L^2 \sum_{i \in A \cap A} (V_{i,1} V_{i,2}) s_i - L^2 \sum_{i \in A \cap A} s'_i - \sum_{i \in A \cap A} \sum_{v \in A \setminus A} r_{i,v} \pi'_{i,v}$ were zero, then the signature would be correct, since the product of the partial signatures of servers $j \in A \setminus A$ is equal to

$$U \equiv \frac{m^{d-P}}{\prod_{i \in A \cap A} m^{L^2 s_i - \sum_{j \in A \setminus \{i\}} \text{PRF}_{\sigma_{i,j}}(m) \text{sign}(i-v)}}$$

$$\equiv \frac{m^{d-P}}{\prod_{i \in A \cap A} m^{L^2 s_i - \sum_{j \in A \setminus A} r_{i,j}(m) \text{sign}(i-v)}} \mod n$$

and thus the product of all partial signatures would equal

$$\begin{aligned} U \cdot \prod_{i \in A \cap A} m^{L^2 s'_i} \\ \equiv U \cdot \prod_{i \in A \cap A} m^{L^2(V_1(i)/V_2(i))s_i + \sum_{j \in A \setminus A} r_{i,j}(m)\text{sign}(i-v)} \\ \equiv m^{d-P} \bmod n. \end{aligned}$$

Thus, we can compute exponents a, b such that $g^a \equiv g_1^b \bmod n$, which implies that we can factor (and thus break RSA).

Similar to the proof of security, we can conclude that if the probability of breaking RSA is negligible and one can break the system, then PRF is not a pseudorandom function family, since the simulation can be made into a polynomial time distinguisher between the a function from PRF and a truly random function.

Implementation consideration: Because of the added inefficiency burden we recommend that the system is run without robustness testing until misbehavior is detected (i.e., an invalid signature is produced). At that point, the robustness enhancements can be incorporated to determine misbehaving servers.

5.2 Proactiveness: Intermixing for Efficiency, No-interaction and Added-Control

There are a number of ways intermixing can help in proactive RSA systems.

1. We may engage in a full proactive update as in [FGMY2] but use pseudorandom intermixing to reduce communication. In addition to share resharing, the pseudorandom functions should be replaced by new ones during updates.
2. We can also employ some dynamic updates which are less costly. One such change is updating the pseudorandom functions only. This can be done interactively (authenticated Diffie Hellman key exchange) between the parties.
3. Proactivization can be used to dynamically add and remove parties (via an update). If we stick to adding and removing from the recently updated group of parties, we can do it also by deciding to employ/not-employ the intermixings shared with them. This is an access-control function which computes on keys (analogous to “control vectors” acting on DES keys [M]). It assures that limitations on the cooperation can be easily achieved with intermixings (using them as credentials).
4. Whereas full proactive refreshment of cryptographic tools is needed to assure that past corruptions (memories learned in the past) are forgotten (namely erased and become irrelevant), we can take “simpler” mechanisms to assure that future corruptions cannot learn the past. This is done by “forward refreshment” of the keys for intermixing. This will ease the simulation arguments as the “pseudorandom past” becomes random. This can be achieved by updating the pseudorandomness based on “current round information”

and in a non-interactive fashion. A tag (i.e., date, counter which can be agreed upon) and previous randomness is used to generate a new pseudorandomness for intermixing followed by an update. This can sometimes be extended to a full proactive update and implies, for example:

Theorem 5. *Using the above technique of forward refreshment with [FGMY] we are able to achieve fault-free non-interactive (e.g., all parties are honest and active in a round) full proaktivization.*

This is the first time that such non-interactive maintenance is possible. It can be derived from [FGMY] (the first proactive RSA solution based on families of committees) by sharing pairwise intermixing where in each committee in a family one adds and the other subtracts using their new locally refreshed pseudorandomness (which is derived from the old key applied to a global tag which can be the global public state of this round). The new intermixing keys generate new shares, which is then followed by a non-interactive verification. At first glance this looks impossible since the adversary moves around in the system. But, notice that an adversary moves out of a server after it is detected in the mobile adversary model (otherwise silent spreading without limits is possible). Thus, when it is detected it is accounted for as a fault which, in turn, causes an interactive refresh which disconnects the past (due to perfect forward secrecy etc.). This amortization of “interaction” against “faults” enables the proof of the Theorem.

Let us give a simple example based on [F89], in which the secret key $d = s_1 + \dots + s_t$ (it is used as a procedure for a “family” in [FGMY]). It demonstrates how proactive update can be done where if there are no faults the update is non-interactive. The new shares now become $s'_i = s_i + \sum_{j=1..i-1,i+1..t} \text{sign}(i-j) \text{PRF}_{\sigma_{i,j}}(\text{tag})$ (we factor in the available “honest” pseudorandomness applied to a current “tag”). Before changing the s'_i a signature for some tag can be tested with the new shares. For [FGMY] there are many such sets s_1, \dots, s_u (held by a family of servers) such that they sum up to d and, moreover, more than one server can possess s_i . Proactive update in this system is a “sum-to-sum” process which takes the l -out-of- l to a different such system (within a family). For robustness of the update, a commitment to $\sigma_{i,j}$ is published as before. Moreover, the distributor (a single dealer or a distributed one) had published g^{s_i} . Publication of commitment to $\text{PRF}_{\sigma_{i,j}}(\text{tag})$ is provided by i (and j) using say $C_{i,j} \equiv g^{\text{PRF}_{\sigma_{i,j}}(\text{tag})}$ (one broadcast, to help update – but no more interaction is needed). Entities i and j may now have a dispute in case they disagree and value is opened if necessary (one will be wrong and removed, we will need to update from a different family if an entire committee is found corrupt). Each i also publishes $g^{s'_i}$ (for the next round using his share) and the following verification is made by each v (within a family with dispute phase if necessary) $g^{s'_i} \equiv g^{s_i} \prod_{j \in \Lambda \setminus \{i\}} (g^{\text{PRF}_{\sigma_{i,j}}(\text{tag})})^{\text{sign}(i-j)}$. If no dispute s'_i is now used.

6 Conclusions

A new technique that entangles algebraic computations with shared pseudorandomness and which applies to numerous existing protocols has been described. We have shown that it can be a design tool for incrementally achieving/ retrofitting correctness, efficiency, provability, and extended functionality in distributed cryptographic protocols. It also allows for better isolation (individualization) of computations and added (access)-control. Inclusion of the tool in various other distributed cryptographic protocols may result in further improvements.

References

- [B97] D. Beaver, Commodity-based Cryptography, In the 29-th STOC, 446–455, 1997. [309](#)
- [BR93] M. Bellare, and R. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, ACM Conference on Computer and Communications Security, 1993. [318](#)
- [BR94] M. Bellare, and R. Rogaway, *Optimal Asymmetric Encryption*, Eurocrypt 94. 92-111.
- [Bl] R. Blakley, *Safeguarding Cryptographic Keys*, FIPS Con. Proc (v. 48), 1979, pp. 313–317. [307](#)
- [BM] G.R. Blakley and C. Meadows, *Security of Ramp Schemes*, Crypto 84, LNCS 196, 242–268. [308](#)
- [BF97] D. Boneh and M. Franklin, *Efficient Generation of Shared RSA Keys*, Crypto 97 proceedings. [315](#)
- [B88] C. Boyd, *Digital Multisignatures*, IMA Conference on Cryptography and Coding, Clarendon Press, 241–246, (Eds. H. Baker and F. Piper), 1989. [307](#)
- [D92] Y. Desmedt. Threshold cryptosystems. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology—AUSCRYPT ’92*, volume 718 of *Lecture Notes in Computer Science*, pages 3–14. 1992, Springer-Verlag. [307](#)
- [DKLMQW] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestreacute, J.-J. Quisquater and J.-L. Willems, *A Practical Implementation of the Timing Attack*, Cardis ’98. [317](#)
- [DDFY] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, *How to Share a Function Securely*, ACM STOC ‘94, pp. 522–533. [307](#)
- [DF91] Y. Desmedt and Y. Frankel, *Shared Generation of Authenticators and Signatures* Advances in Cryptology-Crypto ’91, pp. 457-469. Springer-Verlag. [307](#), [308](#), [312](#), [313](#)
- [DH] W. Diffie and M. Hellman, *New Directions in Cryptography* , IEEE Trans. on Information Theory 22 (6), 1976, pp. 644-654. [310](#), [314](#)
- [F] P. Feldman, *A Practical Scheme for Non-Interactive Verifiable Secret Sharing*, FOCS ’87, pp.427-437. [308](#)
- [FS] A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, Crypto’86, pp. 186-194. [318](#)
- [F89] Y. Frankel, *A practical protocol for large group oriented networks*, In J. J. Quisquater and J. Vandewalle, editor, *Advances in Cryptology, Proc. of Eurocrypt ’89, (Lecture Notes in Computer Science 773)*, Springer-Verlarrg, pp. 56-61. [307](#), [322](#)

- [FY98] Y. Frankel and M. Yung. Distributed public-key cryptosystems. In H. Imai and Y. Zheng, editors, *Advances in Public Key Cryptography—PKC '98*, volume 1431 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, Feb. 1998. invited talk. [307](#)
- [FGMY] Y. Frankel, P. Gemmell, P. MacKenzie and M. Yung. *Proactive RSA*, crypto 97. [307, 322](#)
- [FGMY2] Y. Frankel, P. Gemmell, P. MacKenzie and M. Yung. *Optimal Resilient Proactive Public-Key Systems*, FOCS 97. [307, 308, 312, 314, 315, 316, 317, 319, 321](#)
- [FGY] Y. Frankel, P. Gemmell and M. Yung, *Witness Based Cryptographic Program Checking and Robust Function Sharing*. STOC96, pp. 499–508. [307, 317, 318](#)
- [FMY] Y. Frankel, P. MacKenzie and M. Yung. *Robust Distributed Efficient RSA-key Generation*, manuscript. [315](#)
- [GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Robust Threshold RSA*, Crypto96, pp. 157–172. [307, 308, 317, 318](#)
- [GI99] N. Gilboa and Y. Ishai, *Compressing Cryptographic Resources*, Crypto99, pp. 591–608. [309](#)
- [GGM] O. Goldreich, S. Goldwasser and S. Micali, *How to construct random functions*, J. Comm. Sci. 28 (1984), pp. 270–299. [310](#)
- [HWKS] C. Hall, D. Wagner, J. Kelsey, and B. Schneier, *Building PRFs from PRPs*, Proceedings of Crypto '98, 1998, Springer-Verlag, pp370–389. [310](#)
- [HJJKY] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, *Proactive Public-Key and Signature Schemes* Proceedings of the Fourth Annual ACM Conference on Computer and Communications Security, CCS '97.
- [K] P. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSA and Other Systems*, Crypto96. [317](#)
- [KJJ] P. Kocher, J. Jaffe and B. Jun, *Differential Power Analysis*, Crypto99. [317](#)
- [M] S. M. Matyas, *Key processing with control vectors*, Journal of Cryptology, 3 (2), pp. 113–136, 1991. [321](#)
- [OY] R. Ostrovsky and M. Yung, *How to withstand mobile virus attacks*, Proc. of the 10th ACM Symposium on the Principles of Distributed Computing, 1991, pp. 51–61. [307](#)
- [R] T. Rabin, *A simplified approach to Threshold and Proactive RSA*, Proceedings of Crypto 98, Springer-Verlag, 1998, pp. 89–104. [307, 308, 311](#)
- [RSA] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signature and Public Key Cryptosystems*, Comm. of ACM, 21 (1978), pp. 120–126. [307](#)
- [Sch] C. P. Schnorr, *Efficient identification and signatures for smart cards*, Crypto'89, pp. 239–252. [318](#)
- [Sh] A. Shamir, *How to share a secret*, Comm. of ACM, 22 (1979), pp. 612–613. [307, 315](#)
- [Sho] V. Shoup, Personal communication. [307](#)

A Brief Description of Desmedt-Frankel

The following one-time setup is performed.

- The dealer generates an RSA public key (e, n) and the associated private key d . The modulus n is generated such that it is a composite of two safe primes, $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes. Let $R = 2p'q'$.

- The dealer chooses a random polynomial $f(x) = A_0 + A_1x + \cdots + A_{t-1}x^{t-1}$, such that $A(0) = d - 1$ and $A_i \in_R Z_R$.
- Each Server i is assigned interpolation point $x_i = 2i$ and receives share

$$s_i \equiv f(x_i) \cdot \left(\prod_{j \in \{1, \dots, l\} \setminus \{i\}} (x_j - x_i) \right)^{-1} \pmod{R}$$

over a private channel.

A set of servers Λ can sign a message m by participating in the signing protocol in Figure 3, with $P = 1$, $\text{PRF}_{\sigma_{i,j}}(m) = 0$ for all $i, j \in \Lambda$, and $z_{i,\Lambda} = (\prod_{j \in \{1, \dots, l\} \setminus \Lambda} (x_i - x_j))(\prod_{j \in \Lambda \setminus \{i\}} (0 - x_j))$.

RSA-Based Auto-recoverable Cryptosystems

Adam Young^{*} and Moti Yung^{**}

Abstract. The deployment of a “public-key infrastructure” (PKI) has recently started. Another recent concern in business and on the national level is the issue of escrowed encryption, key recovery, and emergency access to information (e.g., in the medical record area). Independent development of a PKI and an escrowed PKI (whenever required or desired) will pose a lot of constraints, duplication efforts and increased costs of the deployment. It will introduce inter-operability issues which will be hard to overcome. Thus, what we advocate here is a joint design of an escrowed PKI and a regular PKI.

In this work we develop an approach to such an integrated design. We give the first auto-recoverable systems based on RSA (or factoring), whereas the original auto-recoverable auto-certifiable schemes were based on Discrete Logarithm based keys. The security proof of our system assumes only that RSA is hard, while the original schemes required new specific discrete log based assumptions. We also put forth the notion of “generic” auto-recoverable systems where one can start with an unescrowed user key and then by simply doing “re-registration”, change the key into an escrowed one. In contrast, in the original systems the user keys were tightly connected with the escrow authorities’ key. Besides this novel (re)-registration procedure there are no changes or differences for users between a PKI and a generic auto-recoverable PKI.

1 Introduction

In this paper we develop an escrowed public key infrastructure that is as close as possible to infrastructures without escrow, and that can be used to escrow a subset of users. This is useful for numerous reasons. First, showing that an auto-recoverable escrowed PKI (i.e., one that is compatible with a regular PKI) can be based solely on the same functions that are used to implement a PKI (e.g., the RSA function [RSA78]), while at the same time not introducing new mathematical assumptions, makes the system more trustworthy. Second, due to the similarity of the escrowed system with unescrowed ones, much of the same software and practices for the former can simultaneously be applied to the latter, making the composed solution more economical. Lastly, the coexistence of escrowed keys with unescrowed ones within the same infrastructure will minimize the use of escrow to that of the “required keys”, and will enable operators of the PKI to comply with regulations and needs while allowing as much privacy as possible.

Enabling after the fact escrow where an unescrowed key can be used online and then can later be transitioned into “escrowed mode” is a facility which

^{*} Currently: Computer Science, Columbia University. a.young@cs.columbia.edu.

^{**} Currently: CertCo NY, USA. moti@certco.com, moti@cs.columbia.edu

seems useful. It combines on-line privacy with the ability to archive data for “future historical purposes” with controlled access. In this context information and keys can be securely deposited into a national archive. It gives maximal on-line privacy which is a must (for secrecy of operations on the national level). Yet, it should not prevent future access when the accessibility of the data becomes more important than outdated information secrecy. This separability of keys and escrow authorities also implies that the same key can be deposited with different escrow authorities.

The development of a system under the constraint of being “auto-recoverable” [YY98] enables the system to be software-based (i.e., user’s programs can be distributed and operated in software without the need for tamper-proof hardware). Another inherited property due to these constraints is having a key escrow system with minimal overhead (i.e., in the PKI context, making it as close as possible to a regular PKI) while retaining flexibility (e.g., separating users from the inner workings of and access to, escrow agents). This enables the escrowed PKI to be used with the same ease as a typical unescrowed PKI. A list of privacy, abuse-freeness (no leakage), flexibility, and operational properties desired was given in [YY98].

The main challenge in implementing generic systems with a “drop-in replacement” property is basing it on general public-key (trapdoor) functions in a way which achieves certain verification, compliance, and security requirements, and having it such that the security totally relies on the hardness assumptions of the basic cryptographic functions. Along the way we also define the notion of a “zero-knowledge proof of knowledge in the random oracle mode” (we need it for our proofs and we did not find it in the literature). Additional properties we achieve for the system constructed here are:

1. **Employing RSA Keys:** The systems can be based on the most widely deployed system, i.e., RSA/factoring-based systems.
2. **Minimal User Overhead:** The only change for the user is additional information sent during key registration (or re-registration). Users need not re-key to escrow existing public keys.
3. **No Cascading Changes:** The users do not change their applications and systems software which employ cryptography (besides the registration procedure). Not even the cryptographic engine (e.g., an RSA based engine) need be changed.
4. **Separation of Users and Escrow Agents:** The escrow authorities are managed and constructed independently of the users (only their public key(s) need be known for (re)-registration, but not for the user’s key generation).
5. **Independent User Keys:** The user’s key is independent of any other key and is produced in much the same way as in an unescrowed PKI. The users employ the same basic crypto algorithms (key generation, encryption, etc.).
6. **Multiple Escrow Authorities** Users can register for escrow with multiple escrow authorities or with one of their own choosing, among other options.
7. **Granularity of Escrow:** When the escrow agents are activated, they may either open keys or open information encrypted under the keys. This enables

escrow within a specific context while keeping users' privacy otherwise. For national law enforcement applications, this feature is very important to allow minimization of escrow by avoiding revealing the private keys of receivers when senders are the ones under suspicion.

8. **Escrow/non-Escrow coexistence:** The same PKI can handle escrowed and unescrowed keys (which minimizes escrow to the keys needed to undergo the escrow process).
9. **Escrow Hierarchy:** A multi-level security system can be implemented where the escrow authorities at each level can access all of the information below in the hierarchy, and none of the rest of the information.

2 Background and Related Work

The definitions of the notions we employ are given in appendix A. In the appendix we also present the new definition of a zero-knowledge proof of knowledge in the random oracle model (which we need in our proofs and which was not available in the literature).

Micali used VSS to design public key cryptosystems with escrowed capabilities when he proposed the notion of Fair Public Key Cryptosystems [Mi92]. His system employs a “verifiable secret sharing protocol” to distribute users' keys to the escrow agents [CGMA,Fe85]. The problem with Fair PKC's is that first, every user must split his private key and interact with the escrow agents who then need to interact among themselves to approve a key and be convinced that it is escrowed properly. This is *not* a “minimal change” to a PKI (the entire initialization of the system is changed). Second and even more importantly, there are information leakage attacks on such systems. In fact, if used where each user's public key is publicly verifiable, the system requires the publication of an excessive amount of information; this can be abused to permit ‘shadow public keys’ to be published (which are unescrowed), a notion due to Kilian and Leighton [KL95]. The notion of having an escrow system be ‘shadow public key resistant’ is a very recent and important new problem to consider (naive attempts to associate auto-escrow with merely publicly verifiable encryptions or shared encryptions may result in similar information leakage flaws). It is hard to prove shadow public key freeness, but we can require that the public information in an escrowed system be the same as the public information in a regular PKI. Kilian and Leighton [KL95] suggested a correction to Fair PKC's which they call Fail-Safe Key Escrow to avoid shadow public key abuse, but their impractical solution requires even more protocol interaction than Micali's solution.

A more recent solution which attempts to minimize the impact on users when adding escrow and which limits information leakage is the notion of “auto-recoverable auto-certifiable” systems of [YY98], where users interact with CA's only to set up their keys and thereafter use the escrowed PKI as if it were a regular PKI. The system employed double decker exponentiation ([St96] and also [CS97,YY98,YY99]). The starting point model of the solution we present is also that of an auto-recoverable system. We present extended advantages of the notion.

Publicly Verifiable Secret Sharing (PVSS) is related (but not sufficient) for the above notion. A PVSS for sharing a composite of two primes was given in [FO98]. The solution encrypts the user's shares using RSA, and are thus not semantically secure encryptions. The scheme relies on a new modified RSA assumption. Also, when the shares are recovered, a factor of n is obtained, thus no mechanism for decrypting individual messages is given.

A number of recent suggestions have appeared after the initiation of this area in [YY98] and after the initial announcements of our results (e.g. [Man97]). Naturally, we welcome other works in this area. One work is another PVSS for composites which was given in [BT99]. This solution utilizes numerous SZK proofs and requires the use of a composite whose factorization is unknown to all of the parties involved. The solution has the property that it requires $O(2^{21})$ modular exponentiations to recover the shared secret (a factor) of the user, and requires $O(2^{80})$ tries to forge the PVSS proof. Shanks algorithm must be used to recover the shared secret. Like [FO98], the recovery algorithm reveals one of the factors of n , and no algorithm for decrypting individual messages is given; in addition, the encryption is not semantically secure. In [FPS99] an auto-recoverable solution for composites is promised. A cryptographic primitive called Diophantine Commitment (related to bounded range SZK proofs, see [CFT98]) is discussed. The solution is also based on [PS00]. From what is currently available [PS-L], we believe (but may be wrong) that the scheme proposed may not have all the properties we discuss herein, yet it provides very short certificates of recoverability which is a very elegant and useful property. Other schemes related to the original work [YY98] have appeared in [YY99,Ve00] where a scalable security for the recovery agent was provided, and in [Sch99] which simplified the original scheme's proofs.

It is worth while noting that numerous independent works have recently recognized the role of an off-line independent third party. The work on group signatures and ID-escrow [CS97,KP98], escrowed e-cash [CMS96,FTY96], registered mail [Mi92], and fair exchange regarding signatures [ASW98,Ch98], all employed mechanisms for verifying and transferring information to an off-line third party.

3 Basics of a Generic Auto-recoverable PKI

3.1 Defining an Auto-recoverable PKI

Assume that we have a public key function. This function has a key generation procedure GEN, which generates a public key K_1 and its associated private key portion K_2 . Given the pair (K_1, K_2) , when K_1 is registered in the public file, the user can run EMP where they employ the public key for encryption.

Our definition adds components that will be added to a PKI.

Definition 1. *A Generic Auto-Recoverable Cryptosystem contains the following algorithms (CER, VER, REC) such that:*

1. *CER is a publicly known poly-time probabilistic algorithm that takes the output of GEN, which is (K_1, K_2) , and the keys of the escrow authority and generates the triple (K_1, K_2, P) which is left on the tape as output. Here K_2 is the randomly generated private key and K_1 is the corresponding public key. P is the transcript of the zero-knowledge proof of knowledge (its non-interactive version may be a poly-sized certificate that proves that K_2 is recoverable by the escrow authorities using P).*
2. *VER is a publicly known poly-time deterministic algorithm that takes (K_1, P) on its input tape and returns a boolean value. With very high probability, VER returns true iff P can be used to recover the private key K_2 .*
3. *REC is a private poly-time deterministic algorithm that takes (K_1, P) as input and returns K_2 on its tape as output with overwhelming probability provided $VER(K_1, P)$ is true (REC may be a distributed protocol among distributed entities).*
4. *It is intractable to recover K_2 given K_1 and P without REC.*

We say that CER is a valid certification protocol if:

- the protocol: (CER,VER) is a computational zero-knowledge proof of knowledge of K_2 on input K_1 (of size h) (we will describe the non-interactive random oracle based system, but an interactive procedure is also possible).
- In addition the following property holds:
Transferability with error k : For any CER' if $p(h)$ is the probability that the CA accepts in VER on input K_1, P from CER', then REC is successful on the same input with probability at least $\max\{p(h) - k(h), 0\}$, for some $k < p$ (we are interested in negligible k).

Let us next explain in what sense the above is generic. In a regular PKI the following is done: (1) the CA publishes its parameters, (2) using GEN a user generates a key pair, accesses the CA, and registers the public key, (3) using EMP the users employ the system to send encrypted messages. In the generic escrowed PKI system, the CA parameters will include the Escrow Authority (EA)'s shared public key. In registration the user will execute GEN and will execute CER to add to the key the transcript P (or by interaction), the CA verifies it using VER. Otherwise, everything else is as in a regular PKI. Users send messages using EMP as before. When keys are taken out of escrow, the CA needs to cooperate and it supplies P to the EA which can then recover the private information using REC.

3.2 The Basic Structure of the System

To implement the system we will employ the following ingredients:

- General semantically-secure public key cryptosystem of the EA's.
- Public commitment schemes which are homomorphic, so that multiplications of commitment values result in a commitment of the sum of the plaintexts.
- The fact that in the number-theoretic public-key systems there exist homomorphic structure which enables homomorphic commitments.

3.3 System Initialization

The following are the cryptographic primitives that are used in our system by the escrow authorities. ENC is a semantically secure probabilistic public key encryption algorithm that takes three arguments, r , s , and E . Here r is a message to be encrypted, s is a randomly chosen string to make the encryption probabilistic, and E is a public key. Thus, $C = ENC(r, s, E)$ is the ciphertext of the message r . Let DEC be the corresponding decryption function. Thus, $r = DEC(C, D)$, where D is the private key corresponding to E . It could be that D is shared distributively, in which case $r = DEC(C, D_1, D_2, \dots, D_m)$.

Semantic security is sufficient in our case (and there is no need for chosen ciphertext security), since the decryption will be performed to recover keys whose “ciphertext transcript” includes a proof of knowledge of the key by the encrypting party. Note that P proves message awareness.

The system is independent of the organization of the escrow agents as a distributed entity. In the case of m escrow authorities, they generate the shares D_1, D_2, \dots, D_m of their shared private key D , and they collaboratively compute their corresponding public key E . To this end, the notion of function sharing to enable threshold decoding can be employed [DDFY,FGY,GJKR,FGMY].

E is published as the key of the EA’s. We may also allow a number of keys to be published where users choose the escrow authority key or keys to work with.

4 Generic Auto-recoverable Systems

4.1 Constructing the Certification and Recovery Mechanisms

We combine the notions of encryption and zero-knowledge proofs and add them on top of the key generation procedure. For efficiency of communication (which assures that the new scheme is embeddable in an old protocol), we employ the methodology of employing random oracles. This methodology was put forth by Bellare and Rogaway [BR94] to model the use of a cryptographic hash function. This notion was used originally by Fiat and Shamir in [FS86] and also Schnorr [Sc91], and was proved rigorously by Pointcheval and Stern [PS96]. The proof in the random oracle model validates the design and does not constitute a rigorous complexity-theoretic proof, however it typically gives efficient solutions. In our case it reduces interaction. Some weaknesses (not in the current use) are known, and validation via a random oracle is not universal, especially if the cryptographic functions themselves rely on the oracle [CGH98] – but here as well as in many other uses, separation exists. In any case, interactive registration variants of our constructions (which use only two full rounds) exist, and the penalty is only extra interaction. Rather than self-challenging by an oracle as in the descriptions below, a traditional challenge-response interaction takes place. Zero-knowledge proofs in the random oracle model were carefully defined in [BR94].

We will employ “split encryption” where a value is put in a number of places. Such a primitive and its demonstrated power have been used and shown

in various contexts: zero-knowledge proofs, e-cash and secret sharings., e.g., in [IY86,KMO89,CFN,FY93,Fe85,BG96], to give just a partial list.

4.2 RSA Based Systems

CER: Key Certification The user generates a product of two primes $n = pq$. Let d be the inverse of $e \bmod \phi(n)$ (this is necessary, since we need $Z_{\phi(n)}$ to be indistinguishable from Z_n for ZK). Here e and d are the public and private RSA exponents, respectively. We assume that before this protocol is engaged, P proves that n is the product of two primes. A protocol that proves that n is of the form $n = p^r q^s$, where p and q primes (congruent to 3 mod 4 where r and s are both odd) is given in [GHY] (resp. [GP87]). A protocol by Boyar et al. can be used to prove that n is square free [BFL91]. Taken together they assure that n is the product of two primes. There are non-interactive versions of such compliance proofs which we employ. In addition, care must be taken not to allow the exploitation of the “Desmedt subliminal channel” [De88] (as in [YY96]) to leak information. So, the upper half of the bits of n should be generated by a call to a random-oracle hash function and the preimage of this call should be given to the CA for verification.

4.3 Non-interactive Solution

In the protocol that follows, the prover uses an ideal hash function (assumed to be indistinguishable from a random oracle) to generate the values for t'_{i-1} . Thus, the underlying atomic protocol can be viewed as a protocol with a success probability of $\tau(n)/2n$, where $\tau(n)$ denotes the number of elements modulo n with order $\lambda(n)$. Let $k(m) = \omega(\log m)$ be given. We will increase the size of the transcript by a factor of δ to achieve an error of at most $2^{-k(m)}$. In [LS] it was shown that if n is the product of two primes, then $\tau(n) > n/(3 \ln \ln n)$. Since we insist that P proves to V that n is the product of two primes, we can take $\delta = 12(\ln \ln n) \ln 2$ to insure that d is transferred in CER with overwhelming probability.

CER: Key Certification The following is how the non-interactive proof of knowledge transfer P is constructed:

1. $P = (n), t_0 = H(n)$ (H is a random hash function with range Z_n^*)
2. for $i = 1$ to $\delta k(m)$:
3. $t'_{i-1} = H(t_{i-1})$
4. $t_i = t'_{i-1}^e \bmod n$
5. for $i = 1$ to $\delta k(m)$ do
6. $a_i \in_R Z_{\phi(n)}$, choose $s_{i,1}, s_{i,2}$ randomly for use in ENC
7. $v_i = t_i^{a_i} \bmod n$
8. $C_{i,1} = ENC(a_i, s_{i,1}, E), C_{i,2} = ENC(d - a_i \bmod \phi(n), s_{i,2}, E)$
9. add $(v_i, C_{i,1}, C_{i,2})$ to the end of P

10. $\text{val} = H''(P)$ (H'' is a random oracle hash)
11. set $b_1, b_2, \dots, b_{\delta k(m)}$ to be the $\delta k(m)$ least significant bits of val , where $b_i \in \{0,1\}$
12. for $i = 1$ to $\delta k(m)$ do
 13. let $a_{i,1} = a_i$ and let $a_{i,2} = d - a_i \bmod \phi(n)$
 14. add $z_i = (a_{i,j}, s_{i,j})$ where $j = 1 + b_i$ to the end of P

Thus, $P = (n, (v_1, C_{1,1}, C_{1,2}), \dots, (v_{\delta k(m)}, C_{\delta k(m),1}, C_{\delta k(m),2}), z_1, \dots, z_{\delta k(m)})$. Note that since $t_0 = H(n)$, the prover must commit to the composite before the oracle gives the prover the randomly chosen bases $t_1, t_2, \dots, t_{\delta k(m)}$. Thus, a user is able to pick from a polynomial number of $(\delta k(m) + 1)$ -tuples $(n, t_1, t_2, \dots, t_{\delta k(m)})$ with randomly chosen values for t when deciding on the public key to give to the CA. Note that a prover conditioning the transcript for a favorable tuple is the same situation as a prover conditioning a transcript for favorable challenge bits.

VER: Public Escrow Verification The verifier computes $t_1, t_2, \dots, t_{\delta k(m)}$ himself based on n , and the verifier computes $b_1, b_2, \dots, b_{\delta k(m)}$ in the same way as in the certificate generation process. The verifier checks that all of the values in P lie in the correct sets. For example, the verifier checks that the $t_i \in Z_n^*$, and that $a_{i,1+b_i} < n$ for $1 \leq i \leq \delta k(m)$. If any of these verifications fails, then the verifier concludes that the private key is not properly escrowed. For i ranging from 1 to $\delta k(m)$, the verifier verifies the following things:

1. $C_{i,1+b_i} = \text{ENC}(a_{i,1+b_i}, s_{i,1+b_i}, E)$
2. $t_i^{a_{i,1+b_i}} = (t'_{i-1}/v_i)^{b_i} v_i^{1-b_i} \bmod n$

The verifier concludes that the private key is escrowed as long as all the verifications pass and as long as both criterion are satisfied for $1 \leq i \leq \delta k(m)$.

REC: Key Recovery The escrow authorities recover the user's private key as follows. For i ranging from 1 to $\delta k(m)$, the authorities compute d'_i to be the sum of the plaintexts corresponding to $C_{i,1}$ and $C_{i,2}$. Let $K_i = ed'_i - 1$. The escrow authorities then utilize the well known Las Vegas algorithm that factors n given a multiple of $\lambda(n)$. This algorithm is run on K_i for each i .

4.4 Security

To prove the security, we will present the atomic version of the proof in the interactive case. We will then argue its security, and apply the Bellare-Rogaway reduction to prove the security of CER. Let $t' \in_R Z_n^*$ be an element chosen jointly by P and V. P and V compute $t = t'^e \bmod n$.

1. P chooses $a \in_R Z_{\phi(n)}$ and s_1, s_2 randomly for use in ENC
2. P computes $v = t^a \bmod n$
3. P computes $C_1 = \text{ENC}(a, s_1, E)$, $C_2 = \text{ENC}(d - a \bmod \phi(n), s_2, E)$
4. P sends (v, C_1, C_2) to V

5. V sends $b \in_R \{0, 1\}$ to P
6. P sends $z = (w, s_{1+b}) = (a^{1-b}(d-a)^b \text{ mod } \phi(n), s_{1+b})$ to V
7. V verifies that $C_{1+b} = \text{ENC}(w, s_{1+b}, E)$ and $t^w = (t'/v)^b v^{1-b} \text{ mod } n$

Lemma 1. *The atomic 3-round interactive protocol is computational zero-knowledge.*

Proof. To prove that it is ZK, we will give a description of a poly-time simulator S^* that is capable of producing transcripts for the interactive version that are polynomially indistinguishable from a transcript derived from a prover and verifier in an interactive session. Let V^* be any polynomial-time probabilistic algorithm that a (possibly cheating) verifier uses to generate his challenges. S^* is defined as follows:

1. $t = t'^e \text{ mod } n$
2. $b' \in_R \{0, 1\}$, $w, w' \in_R Z_n$, choose s_1, s_2 randomly for use in ENC
3. $v = t^{w(1-b')}(t'/t^w)^{b'} \text{ mod } n$
4. $C_{1+b'} = \text{ENC}(w, s_{1+b'}, E)$, $C_{2-b'} = \text{ENC}(w', s_{2-b'}, E)$
5. call V^* with input (v, C_1, C_2) obtaining challenge b
6. if $(b' = b)$ then return $(v, C_1, C_2, b, w, s_{1+b})$ else goto 2

Note that $w, w' \in_R Z_n$ in the simulation, since the simulator does not know $\phi(n)$. However $Z_{\phi(n)}$ and Z_n are statistically indistinguishable. Note that indistinguishability won't hold if there exists a line tapper T_N that when given $\{n, v^T, C_j^T, v^F, C_j^F\}$ (in no particular order) can distinguish C_j^T as containing, for example, the plaintext $(d-w)^T$ from C_j^F containing $(d-w)^T$ with probability $> 1/2 + N^{-c}$. Here a^T indicates a value a that is in a transcript formed between a prover and a verifier, and b^F indicates a value b that is in a transcript forged by the simulator. The ciphertexts correspond to the unopened ciphertexts, and $j \in \{1, 2\}$. Recall that semantic security implies that anything that can be learned about the plaintexts from these two ciphertexts can be learned without the ciphertexts, so the line tapper is unable to distinguish based on these values (even if T_N were given both plaintexts). Hence, semantic security is a sufficient condition. QED.

Note that adaptive chosen ciphertext security of ciphertexts is not needed in our system since users never see the decryptions of the values in the certificates (yet their proof achieves the notion via the formalism of a proof of knowledge). The interactive version of the CER protocol is a typical 3-round computational zero-knowledge proof. Completeness and soundness are straightforward. It differs from standard zero-knowledge proofs in that, in addition to a commitment value being sent in the first round (i.e., using v to commit to the base t logarithm of $v \text{ mod } n$), the prover sends two semantically secure encryptions (which must be consistent with the commitment in v , and d). The first of these encryptions can be thought of as yet another commitment of the base t logarithm of v . The second encryption is a commitment to the base t logarithm of t'/v . Since these are both

semantically secure encryptions, they give as much information to a poly-time adversary as the adversary can compute himself without the encryptions. They are, in some sense, redundant commitments of v and t'/v (they are useful since they provide third party recoverability). In the second round, the verifier sends a randomly chosen bit to the prover, as in standard zero-knowledge proofs. In the third round, the verifier has to open either a (for v) or $d - a \bmod \text{ord}(t)$ (for t'/v), as in standard zero-knowledge proofs. The only difference is that exactly one of the two semantically secure encryptions (commitments) is also opened, and verified for consistency (to insure third party recoverability).

In section 5.2 of [BR94], a reduction is given that shows how to transform any three move atomic zero-knowledge proof with error probability $1/2$ for $L \in NP$ into a non-interactive ZK proof in the random oracle model. The only significant differences here are that two semantically secure encryptions are sent in each round (this is secure for the same reason as in the DL version) and that the error probability of the non-interactive RSA atomic protocol is not $1/2$. The fundamentals of the reduction and the proofs of soundness and zero-knowledge are unchanged. So, we have the following.

Lemma 2. *The non-interactive CER protocol for factoring based keys is sound and zero-knowledge in the random oracle model.*

Completeness of the non-interactive protocol for factoring based keys is straightforward. We will now prove that the non-interactive CER protocol for factoring based keys is a proof of knowledge in random oracle model. This will shed some light on the differences in the proof of soundness as compared to the soundness discussed in section 5.2 of [BR94]. See the appendix for a formalization of what it means for a non-interactive proof to be a proof of knowledge in the random oracle model. Note that this proof assumes that $b_1, b_2, \dots, b_{\delta k(m)}$ are included in the transcript. This is a minor modification to the protocol, which clearly can be made since the verifier can verify this information.

Lemma 3. *The non-interactive CER protocol for factoring based keys constitutes a proof of knowledge with knowledge error at most $2^{-k(m)}$.*

Proof. It is easy to see that the non-triviality condition holds. We will now consider the validity condition. The common input is $\alpha = n$. We have that $x_i = (v_i, C_{i,1}, C_{i,2})$, $y_i = z_i$, $t = \delta k(m)$, and the b_i 's in CER are the same as the b_i 's in the definition. Suppose that P makes no extra oracle queries. In this case P fools V in round i with probability $(1 - \tau(n)/2n)$ and the probability that P fools V in all $\delta k(m)$ rounds is $(1 - \tau(n)/2n)^{\delta k(m)}$. The knowledge error $\kappa(\alpha)$ in this case equals $(1 - \tau(n)/2n)^{\delta k(m)}$. Note that n being the product of two different primes implies that $\tau(n)/n > 1/(3 \ln \ln n)$, and this implies that $(1 - \tau(n)/2n)^{\delta k(m)} \leq (1 - 1/(6 \ln \ln n))^{\delta k(m)}$. This can be shown to be at most $2^{-2k(m)}$ from the following fact:

$$\text{For all real numbers } t \text{ and } \theta, \text{ s.t. } \theta \geq 1 \text{ and } |t| \leq \theta, (1 + t/\theta)^\theta \leq e^t$$

Now, suppose that P makes $T(m)$ oracle queries (e.g., to try to fix the first several challenge bits to 0). It can be shown that the knowledge error is at most

$T(m)2^{-2k(m)}$, which for sufficiently long m is at most $2^{-k(m)}$. It follows that $p(\alpha)$ is at least $1 - 2^{-k(m)}$.

Let $T = P_{\alpha,\beta,r}(H)$ and $T' = P_{\alpha,\beta,r}(H')$. Consider the following knowledge extractor K . Suppose that in round i , b_i in T is 0 and b_i in T' is 1. K then adds the w_i in T to the w_i in T' to get d_c , a candidate decryption exponent. The operation of K when the bits are inverted is similar.

We will now give a lower bound on K 's probability of extracting a witness from T and T'^1 . Assuming no extra oracle queries are made, with probability $1/2$ we have that b_i in T equals b_i in T' , since the b_i 's are chosen randomly for both T and T' (H and H' serve as honest verifiers). So, with probability $1/2$, the b_i 's in each transcript differ. Also, both transcripts will use the same t_i 's and with probability $\tau(n)/n$, t_i has maximal order. To see this recall that P for both transcripts is using: the same common-input α , the same auxiliary input β , and the same random tape r . Thus, with probability $1 - \tau(n)/2n$ a decryption exponent is not extracted in round i . So, the probability that a decryption exponent isn't extracted in any of the $\delta k(m)$ rounds is $(1 - \tau(n)/2n)^{\delta k(m)}$, which is at most $2^{-2k(m)}$. Now consider the case where P makes $T(m)$ additional oracle queries. It can be shown that the probability that a decryption exponent isn't extracted is at most $T(m)2^{-2k(m)}$. For sufficiently long m this probability is at most $2^{-k(m)}$. Thus, the probability that a valid decryption exponent is extracted by K is at least $1 - 2^{-k(m)}$. Using this worst case value, and the worst case values for $p(\alpha)$ and the knowledge error, it follows that the inequality in the validity condition evaluates to $0 \geq -2^{-k(m)}$. Thus, the validity condition is satisfied. It follows that the non-interactive CER protocol is a proof of knowledge. QED.

Note that all the above steps are efficient (small polynomial overhead, which is reasonable for a one-time operation such as key (re)-registration). So, what we achieve is:

Theorem 1. *There exists a generic and efficient auto-recoverable auto-certifiable system, where the users' keys are based on the RSA/factoring system assuming only the security of RSA/factoring.*

5 Decrypting Individual Messages

In some settings, particularly in law-enforcement settings, it is important to provide the escrow authorities with the ability to decrypt individual messages of users without revealing the users escrowed private key. This is important, since the sender's private key should not be opened if the receiver is under suspicion; this issue is not dealt in systems like the Fair cryptosystems. This mode of operation is possible in our solution by making only minor changes in the system (to the organization of the escrow authorities).

Suppose for example that there are three escrow authorities with three separate semantically secure public encryption functions. The user in this system

¹ We won't derive an exact probability, since using $T(m)$ oracle queries, P may always try to make the first several t_i 's not have order $\lambda(n)$ to trick the prover, for example.

generates three separate public keys, and then escrows each of them using the auto-recoverable auto-certifiable cryptosystem corresponding to each of the public encryption functions of the escrow authorities. The three public keys are verified and published. Suppose that the users use a hybrid cryptosystem to send confidential messages. The sender simply encrypts the session key using all three public keys. The escrow authorities can recover the message without any of them knowing all three of the corresponding private keys.

Alternate methods also exist for composite public keys. One idea is as follows. Now, suppose that there are three escrow authorities. The user can secret split d additively to get d_1, d_2 , and d_3 in conjunction with the auto-recoverable auto-certifiable cryptosystem. It follows that with such additive values for d escrowed, the escrow authorities can perform shared decryption of a message encrypted with n without revealing the user's private key. What we then get is:

Theorem 2. *The generic RSA-based auto-recoverable cryptosystem presented here can be used to decode messages securely by the escrow agents, without revealing the private key of the receiver.*

6 Arbitrary Depth Escrow Hierarchy

Consider a three level key escrow system. The hierarchical tree consists of the escrow authorities at the top, subordinate escrow authorities at the middle level, and users at the bottom. The authorities at the top publish an escrowing public key E . Then each of the subordinate escrow authorities auto-certifies a unique public key using E . Finally, the users assigned to a given subordinate escrow authority generate public keys and auto-certify them using the escrowed public key of their corresponding subordinate escrow authority.

In the special case of the composite based auto-recoverable solution, it is necessary that the modulus at each level fit within the message space of ENC at the level above. We can therefore easily implement a depth 3, 4, or 5 hierarchy in a straightforward fashion.

Theorem 3. *The generic auto-recoverable cryptosystems for factoring/RSA based keys presented here can be used to efficiently implement an arbitrary depth hierarchical key escrow system.*

7 Smaller Certificate of Recoverability

Recall that a safe prime p is a prime of the form $p = 2p_1 + 1$ where p_1 is prime. The CA storage can be reduced as follows. We insist that the user prove that n is the product of two safe primes. This can be done using [CM99]. We remark that this proof can be made non-interactive. Also, in each iteration t_i is computed such that $J(t_i/n) = -1$ by making successive oracle queries. This fact is verified for each t_i by the verifier.

Having done this it then follows that each round in the certificate will transfer d with knowledge error negligibly larger than $1/2$. To see this note that the only

possible orders for t_i are $(2, 2p_1, 2q_1, 2p_1q_1)$ where $q = 2q_1 + 1$. This follows from the fact that t_i is checked by the verifier to be a quadratic non-residue mod n . If the order is $2p_1$ then [ML98] can be used to factor n (the same holds for $2q_1$). This can be seen from the fact that $\gcd((t_i^2 \bmod n) - 1, n)$ is a non-trivial factor of n for such a t_i . If t_i has order $2p_1q_1$ then round i transfers $a + (d-a) \bmod \lambda(n)$. Thus, for all possible orders of t_i , d is transferable in round i (except if $\text{ord}_n(t_i) = 2$ which is an event having negligible probability). This combined with the knowledge error due to the challenge bit, achieves the stated knowledge error of $1/2$. This implies that the number of iterations can be the same as in the NIZK proof in [BR94].

References

- [ASW98] N. Asokan, V. Shoup, M. Waidner. Optimistic Fair Exchange of Digital Signatures. In *Advances in Cryptology—Eurocrypt ’98*, pages 134–148. [329](#)
- [BFL91] J. Boyar, K. Friedl, C. Lund. Practical zero-knowledge proofs: Giving hints and using Deficiencies. In *Journal of Cryptology*, 4(3), pages 185–206, 1991. [332](#)
- [BG96] M. Bellare, S. Goldwasser. Encapsulated Key Escrow. manuscript, 1996. [332](#)
- [BR94] M. Bellare, P. Rogaway. Random Oracles are Practical. In *ACM CCCS ’94*. [331](#), [335](#), [338](#), [340](#), [341](#)
- [BT99] F. Boudot, J. Traore. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *ICICS ’99*. [329](#)
- [Ch98] L. Chen. Efficient Fair Exchange of Verifiable Confirmation of Signatures. In *Advances in Cryptology—Asiacrypt ’98*. [329](#)
- [CFN] D. Chaum, A. Fiat, M. Naor. Untraceable Electronic Cash. In *Advances in Cryptology—Crypto ’88*, pages 319–327. [332](#)
- [CFT98] A. Chan, Y. Frankel, Y. Tsiounis. Easy Come - Easy Go Divisible Cash. In *Advances in Cryptology—Eurocrypt ’98*, pages 561–575. [329](#)
- [CGH98] R. Canetti, O. Goldreich, S. Halevi. The Random Oracle Methodology, Revisited. In *ACM STOC ’98*. [331](#)
- [CGMA] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *FOCS ’85*. [328](#)
- [CM99] J. Camenish, M. Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In *Advances in Cryptology—Eurocrypt ’99*. [337](#)
- [CMS96] J. Camenish, U. Maurer, M. Stadler. Digital Payments Systems with Passive Anonymity Revocation Trustees. In *Esorics ’96*. [329](#)
- [CS97] J. Camenish, M. Stadler. Efficient Group Signatures. In *Advances in Cryptology—Crypto ’97*, pages 410–424. [328](#), [329](#)
- [De88] Yvo Desmedt. Abuses in Cryptography and How to Fight Them. In *Advances in Cryptology—CRYPTO ’88*. [332](#)
- [DDFY] A. De Santis, Y. Desmedt, Y. Frankel, M. Yung. How to Share a Function Securely. In *ACM STOC ’94*, pages 522–533. [331](#)
- [Fe85] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *FOCS ’87*. [328](#), [332](#)
- [FGMY] Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung. Optimal Resilience Proactive Public Key Systems. In *FOCS ’97*. [331](#)
- [FGY] Y. Frankel, P. Gemmell, M. Yung. Witness based Cryptographic Program Checking and Robust Function Sharing. In *ACM STOC ’96*. [331](#)

- [FO98] E. Fujisaki, T. Okamoto. A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications. In *Advances in Cryptology—Eurocrypt ’98*, pages 32–46. [329](#)
- [FPS99] P. Fouque, G. Poupard, J. Stern. Recovering Keys in Open Networks. In IEEE ITW, 1999. [329](#)
- [FS86] A. Fiat, A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology—Crypto ’86*, pages 186–194. [331](#)
- [FTY96] Y. Frankel, Y. Tsiounis, M. Yung. Indirect Discourse Proofs: Achieving Efficient Fair Off-Line Cash. In *Advances in Cryptology—Asiacrypt ’96*. [329](#)
- [FY93] M. Franklin, M. Yung. Towards Provably Secure Efficient Electronic Cash. In *ICALP ’93*. [332](#)
- [GHY] Z. Galil, S. Haber, M. Yung. Minimum-knowledge Interactive Proofs for Decision Problems. In *SIAM J. of Computing*, (4), pages 711–739, 1989. [332](#)
- [GJKR] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Robust and Efficient Sharing of RSA. In *Advances in Cryptology—Crypto ’96*. [331](#)
- [GP87] J. van de Graaf, R. Peralta. A simple and secure way to show the validity of your public key. In *Advances in Cryptology—Crypto ’87*, pages 128–134. [332](#)
- [IY86] R. Impagliazzo, M. Yung. Direct Minimum-Knowledge Computations. In *Advances in Cryptology—Crypto ’86*. [332](#)
- [KL95] J. Kilian, F.T. Leighton. Fair Cryptosystems Revisited. In *Advances in Cryptology—Crypto ’95*, pages 208–221. [328](#)
- [KMO89] J. Kilian, S. Micali, R. Ostrovsky. Minimum-Resources Zero-Knowledge Proofs. In *FOCS ’89*. [332](#)
- [KP98] J. Kilian, E. Petrank. Identity Escrow. In *Advances in Cryptology—Crypto ’98*, pages 169–185. [329](#)
- [LS] M. Liskov, R. D. Silverman. A Statistical Limited-Knowledge Proof for Secure RSA Keys. Submitted to the IEEE P1363 Working Group. Available at <http://grouper.ieee.org/groups/1363/contrib.htm>. [332](#)
- [Man97] A. Young, M. Yung. Manuscript related to the current work dated Sept. ’97 available from the authors (preliminary version also submitted to Eurocrypt ’99.) [329](#)
- [Mi92] S. Micali. Fair Public-Key Cryptosystems. In *Advances in Cryptology—Crypto ’92*, pages 113–138. [328](#), [329](#)
- [ML98] W. Mao, C. H. Lim. Cryptanalysis of Prime Order Subgroups of Z_n^* . In *Advances in Cryptology—Asiacrypt ’98*, pages 214–226. [338](#)
- [PY] P. Paillier and M. Yung. Self-Escrowed Public-Key Infrastructures. (Manuscript).
- [PS96] D. Pointcheval, J. Stern. Security Proofs for Signature Schemes. In *Advances in Cryptology—Eurocrypt ’96*. [331](#), [340](#), [341](#)
- [PS00] G. Poupard, J. Stern. Short Proofs of Knowledge of Factoring In These proceedings. [329](#)
- [PS-L] G. Poupard, J. Stern. Talks at Luminy, Oct. ’99. [329](#)
- [RSA78] R. Rivest, A. Shamir, L. Adleman. A method for obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the CACM*, 21(2), pp. 120–126, 1978. [326](#)
- [Sch99] B. Schoenmakers. A simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. In *Advances in Cryptology—Crypto ’99*, LNCS 1666, 148–164. [329](#)
- [Sc91] C. P. Schnorr. Efficient Signature Generation for Smart Cards. In *Journal of Cryptology*, 4 (3), pages 161–174, 1991. [331](#)

- [St96] M. Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology—Eurocrypt ’96*, pages 190–199. 328
- [Ve00] E. Verheul, Certificates of Recoverability with Scalable Recovery Agent Security. In These Proceedings. 329
- [YY96] A. Young, M. Yung. The Dark Side of Black-Box Cryptography, In *Advances in Cryptology—Crypto ’96*. 332
- [YY98] A. Young, M. Yung. Auto-Recoverable and Auto-Certifiable Cryptosystems. In *Advances in Cryptology—Eurocrypt ’98*. 327, 328, 329
- [YY99] A. Young, M. Yung. Auto-Recoverable Cryptosystems with Faster Initialization and The Escrow Hierarchy. In *PKC ’99*. 328, 329

A Appendix: Definitions

We employ the RSA function and the notion of semantic security in our proposed system. In public-key systems, the security of the encryptions of preimages of public one-way function values is equivalent to the notion of polynomial-security where a challenge of two messages is given, only one of which is the “real message”, and where no one can tell which one is the actual message. In our applications however, we will encrypt a value for which there is a “public commitment” which is related to a public key. In such cases we could not withstand a challenge, since the public commitment is done via a one-way function of the message. However, semantic security still holds. Intuitively, this means that the added encryption does not help beyond what is known from the public commitment.

In [BR94], it was stated that a formalization of what it means for a NIZK proof to be a proof of knowledge was forthcoming. We know of no such formalization to date anywhere in the literature, so we will present a formalization here to prove security in the random oracle model.

To define a “proof of knowledge” in the random oracle model, one can apply the probabilistic notions in [PS96]. Informally, we define an extractor which invokes the prover on two transcripts T and T' , which are initially the same but which are extended identically using random oracles which are only “polynomially” different. That is, the oracle entries used in extending T' are random and with high probability distinct from the corresponding entries in the oracle used to construct T . The forking argument of [PS96] can be cast into an extractor argument: a proof system is a proof of knowledge if there exists a knowledge extractor that when given access to a prover which constructs proofs in both extinctions, is able to extract a witness with probability greater than or equal to the probability of P convincing V that P knows the witness minus the knowledge error.

Definition 2. Denote by $P_{\alpha,\beta,r}(H) = (\alpha, x_1, x_2, \dots, x_t, b_1, b_2, \dots, b_t, y_1, y_2, \dots, y_t)$ the message sent by machine P with common-input α , auxiliary-input y , and random input r when given access to random oracle H . H is random with the restriction that $H(x_1, x_2, \dots, x_t) = (b_1, b_2, \dots, b_t)$. Here x_1, x_2, \dots, x_t , b_1, b_2, \dots, b_t , and y_1, y_2, \dots, y_t are strings. The function $P_{\alpha,\beta,r}$ is called the transcript specification function of machine P with common-input α , auxiliary input β , random input r , and access to a random oracle.

Definition 3. Define the random oracle randomization operation, ROR , to be the following. $ROR(H, x_1, x_2, \dots, x_t) = H'$, where $H, H' \in 2^\infty$ and x_1, x_2, \dots, x_t are strings. H and H' are identical random oracles except that $H'(x_i) \in_R \{0, 1\}^\infty$ for $1 \leq i \leq t$.

We can (w.v.h.p) choose the range values of H' randomly without causing conflicts in the entries which H' and H share since the set from which the range of a an oracle is drawn is uncountable (whereas the tables for H and H' are only countably infinite).

Definition 4. Let R be a binary relation, and let κ be a function from the natural numbers to $[0, 1]$. Denote by $p(\alpha)$ the probability that the machine V accepts on input α , when interacting with the prover specified by $P_{\alpha, \beta, r}$. Let the symbol \perp denote failure to find $\beta \in R(\alpha)$. We say that a function V is a knowledge verifier for the relation R in the random oracle model with knowledge error κ if the following two conditions hold.

1. *Non-triviality:* There exists an interactive function P so that for every $(\alpha, \beta) \in R$, and for all $H \in 2^\infty$, all possible messages sent from P to V on common-input α and auxiliary input β are accepting.
2. *Validity (with error κ):* We say that the verifier V satisfies validity with error κ if there exists a probabilistic expected polynomial-time oracle machine K such that for every interactive function P , every $\alpha \in L_R$, for every $H \in 2^\infty$, it is the case that when K has access to $P_{\alpha, \beta, r}(H)$ and $P_{\alpha, \beta, r}(H')$ such that $H' = ROR(H, x_1, x_2, \dots, x_t)$, K outputs an $s \in R(\alpha) \cup \{\perp\}$, and

$$\Pr\{K \text{ outputs an } s \in R(\alpha)\} \geq p(\alpha) - \kappa(|\alpha|).$$

We call such an oracle machine K a random oracle knowledge extractor. The reader may be wondering why we insisted on using the ROR operation in our definition, since no ROR operation was used in the definition of the forking lemma in [PS96]. In our opinion, this is a minor oversight in the formal definition of the forking lemma. To see this, note that the forking lemma assumes that the machine that replays the oracle machine A (A is a no-message attacker against the signature scheme) replays it with “a different” random oracle. Thus, it is not clear which signature algorithm should be used in practice, since the proof of security assumes access to both signature algorithms (i.e., the signature algorithm with the original oracle and the signature algorithm with the ‘different’ oracle), since clearly both cannot be used to sign a given message.

In the definition of a non-interactive proof being ZK in [BR94], it is made very clear that a random oracle completion operation, ROC , is used to insure that the oracles are *only polynomially different*. As such, we may use one oracle in practice, and the fact that the oracle may have polynomially many “faults” is intractable to detect. By faults we mean entries in the infinite table that are defined to have two values, when only one string from $\{0, 1\}^\infty$ is allowed.

Efficient and Fresh Certification

Irene Gassko¹, Peter S. Gemmell², and Philip MacKenzie³

¹ Bell Laboratories, Lucent Technologies
1600 Osgood Street, N. Andover, MA 01845
gassko@lucent.com

² Computer Science Department, University of New Mexico
Albuquerque, NM 87131 gemmell@cs.unm.edu

³ Information Sciences Research Center, Bell Laboratories
600 Mountain Ave., Murray Hill, NJ 07974-0636
philmac@research.bell-labs.com

Abstract. Electronic commerce is becoming more and more commonplace, but security is still a major concern. To provide security, a good public-key infrastructure (PKI) is needed. However, PKIs have been slow in developing, with one of the major difficulties being the creation of certification authorities (CAs), and in particular, dealing with the problem of certificate revocation. We propose a new solution to this problem. Our solution is based on the idea that individually signed certificates provide little information over any significant time period, given that they may be revoked. That is, after a certain amount of time, a certificate is not useful without some more recent knowledge that it has not been revoked. In all previous work, this has either been handled by off-line/on-line schemes, which require costly updates by the CA for every outstanding certificate for every update period, or by certificate revocation lists/trees.

We propose a system called EFECT (Easy Fast Efficient Certification Technique), which combines the best properties of individual certificates and certificate revocation trees. We show that EFECT allows CAs to be *more secure*, even while providing *more frequent freshness updates* for certificates, and making certification verification *extremely lightweight*. We compare EFECT to previously proposed systems, including traditional X.509 certificates and Certificate Revocation Lists (CRLs), SDSI/SPKI, Micali's Certificate Revocation System (CRS), Kocher's Certificate Revocation Trees (CRTs), and Naor and Nissim's 2-3 Certificate Revocation Trees (23CRTs). Finally, we discuss some novel qualities of EFECT that no previous solution possesses.

1 Introduction

There is an ever increasing need for public-key systems to provide security in network protocols, particularly in electronic commerce. One of the current bottlenecks is the development of a public-key infrastructure (PKI), namely, certification authorities (CAs). In this paper we consider how to implement a large CA, such as for a credit card company.

The defining feature of all currently proposed CA implementations is that they have a digital certificate, consisting of a c-statement (i.e., a *certificate statement* which might include information like a person's name, a public key, and a birth date), along with the CA's signature, thus validating that c-statement. The problem with this is that authorization may be revoked at some future time. Then to maintain security, either certificates have to be issued with very short expiration periods (necessitating frequent reissuing), or there must be a mechanism for certificate revocation. There have been many recent proposals to solve the problem using one or both methods [Micali,Rivest,Kocher,NaorNissim]. We will review these in Section 2.

We propose a system called EFECT (Easy Fast Efficient Certification Technique), which allows us to eliminate structures containing certificate revocation information, such as Certificate Revocation Lists (CRLs) or Certificate Revocation Trees (CRTs) from certificate directories. Thus certificate search alone allows us to find both a certificate and information sufficient for its validation, including current status of a c-statement . This greatly simplifies secure communication. Our model combines the best properties of individual certificates and certificate revocation trees, and possesses some novel properties that no previously proposed solution offers. We show that EFECT allows CAs to be *more secure*, even while providing *more frequent freshness updates* for certificates, and making certification verification *extremely lightweight*.

The design of EFECT is relatively simple. As in previous models, it assumes the existence of *certificate directories*, untrusted databases that store information periodically released and authenticated by the CAs, and make this information available on-line. EFECT uses Merkle Trees [Merkle] like other previously proposed systems, such as CRS2 [Micali], [Kocher], and 23CRT [NaorNissim], but for the purpose of authenticating the certificates themselves, instead of simply certificate revocation information. In EFECT, these trees are called certificate verification trees (CVTs). The untrusted directories store the CVTs, and can thus provide information about the validity of certificates to inquiring parties.

In Section 5, we discuss some of the advantages of this new approach, along with direct comparisons to previously proposed systems.

2 Background

In this section we review previous work in the area of CAs and certificate revocation. In all of these schemes, the CA individually signs each c-statement, and another structure is stored by a directory and used to determine if the c-statement is still valid (i.e., still *fresh*). For concreteness, we assume daily freshness updates.

2.1 X.509 Certificates and Certificate Revocation Lists (CRL)

Certificate Revocation Lists (CRLs) used with X.509 certificates is the standard certificate revocation scheme that is currently being deployed. A certificate is

just a c-statement with serial number, issue date, and expiration date, signed by the CA. A CRL consists of a list of serial numbers of revoked certificates (that have not yet expired) along with a time stamp and some other information, signed by the CA.

In general, after a certificate's signature is verified, to find if the certificate (signed c-statement) is still valid, one would query a directory for a copy of the most recent CRL. To answer a query, the directory must send this CRL in its entirety. Note that once the expiration date of a revoked certificate is reached, its serial number can be removed from the CRL, since the certificate is obviously no longer valid.

2.2 Certificate Revocation Systems

CRS1 CRS1 is one of Micali's [Micali] schemes to solve the certificate revocation problem. It uses an off-line/on-line signature scheme [EGM] to update signatures daily with reduced computation and communication cost (compared to recomputing and redistributing completely new signatures daily). The scheme involves the use of a one-way function f . A random value y_0 is chosen, and the value $f^k(y_0)$ is stored in the certificate, where $f^k(y_0)$ is defined recursively as $f(f^{k-1}(y_0))$. Then the signature may be updated k times, where the i th update involves releasing the value $f^{k-i}(y_0)$. The most recent released value serves as a freshness proof for the certificate.

Once a certificate signature is verified, to find if the certificate is still valid, one would query a directory for a copy of the most recent update value, which can then be checked to see if it corresponds to the value in the certificate. The query communication cost is thus very short. However, the CA-to-directory communication costs are high, since the CA must send the new hash value for every outstanding certificate. Although each hash can be computed relatively quickly, the verification time is proportional to the number of updates since the certificate was issued, which severely limits the lifetime of the certificate and the rate of updates.

Note that most of the following schemes may be combined with CRS1 to allow some communication/freshness tradeoffs.

CRS2 CRS2 is another of Micali's schemes. It uses Merkle Trees to provide information about the revocation status of a certificate. Each serial number of an outstanding certificate is assigned two bits indicating whether it was issued and whether it was revoked. These bits are stored in leaves of the tree (64 certificates, or 128 bits, per leaf), and each parent node in the tree stores the hash of the children's values. Finally the root is signed by the CA.

Once a certificate signature is verified, to find if a certificate is still valid, one could query a directory for the hash values on the path from the certificate's leaf to the root and the signature on the root. Then one would check the appropriate bits at the leaf, verify the hash values, and then verify the root signature. The query communication is logarithmic in the number of certificates, plus the length

of a signature. CA-to-directory communication consists only of changed leaf values and the new root signature. All other hash values can be computed by the directory. (Although not stated by Micali, it is assumed that the signature on the root value also contains a date, so as to prevent replay attacks.)

One disadvantage of this scheme is that it provides extraneous information to the relying party, by informing it not only of revocation status of the certificate in question, but of the revocation status and existence of many neighboring certificates, which is a dangerous security policy and can facilitate many attacks.

2.3 Certificate Revocation Tree (CRT)

CRT is a scheme from Kocher [Kocher] which also uses Merkle trees, but each leaf basically consists of a single revoked certificate. Thus the CRT could be much smaller than the tree in CRS2 (depending on the number of revoked certificates).

Again, it has the features of CRS2, providing good query communication costs. However, it seems that any change might result in recomputation of the entire tree (since one new revoked certificate could change the whole structure of the tree).

2.4 Certificate Revocation 2-3 Tree (23CRT)

Naor and Nissim [NaorNissim] eliminate the problem in Kocher's CRT scheme by replacing the CRT with a 2-3 tree. Then insertion and deletions do not require changing the whole tree, but just a path (logarithmic in the size of the tree).

Notice, however, that the last two schemes are constrained to providing revocation information only, and do not allow a straightforward adaptation for cases when a certificate search has to be provided (e.g. when a CA certificate needs to be retrieved for the purpose of path construction).

2.5 SDSI/SPKI

SDSI/SPKI is a combination of the Simple Distributed Security Infrastructure of Rivest and Lampson [SDSI], and Simple Public Key Infrastructure of Ellison [SDSI]. These were suggested as a simpler alternative to X.509. Revocation (or as they call it, *reconfirmation*) is handled online.

2.6 PayTree

PayTree [JY] is not actually a CA system, but a micro-payment system which involves an entity signing the root of a Merkle tree over random-valued leaves which serve as coins. The idea is that efficiency is gained by not having to sign coins individually.

3 Our System

In their seminal paper, Diffie and Hellman [DH] suggested using a trusted directory representing a modified telephone book which stores public keys (instead of telephone numbers) for secure communications. Kohnfelder [Kohnfelder] noticed that using an on-line service to provide these certificates can create a communications bottleneck. To alleviate this problem he proposed the creation of digitally-signed directory entries which he called certificates. Now certificates could be kept in untrusted directories and retrieved upon requests from users. A big problem with this approach was the possibility of premature revocation. This led to introduction of various structures for storage of revocation information, such as CRLs and CRTs. We suggest a model that eliminates the need for keeping separate revocation data by augmenting a common index structure used for search in directories: B-trees with additional values in each node. This design allows us to use Merkle trees for certificate validation and allows us to eliminate individual signing of certificates. It turns out that the latter step leads to many advantages in our scheme.

In the next subsection we describe the EFEKT model in detail.

3.1 Certification Verification Tree (CVT)

Since the most resource consuming part of the certification process are digital signatures, we abolish the tradition of individually signed certificates.

For all certificates distributed by CA, a B-tree (for concreteness you can think of it as a 2-3 tree) is constructed in the following way: each leaf is a c-statement plus the hash of that c-statement. Any types of c-statements can be used with our model, including those in the popular X.509 format or SDSI/SPKI format.

The hash values of siblings in the B-tree are hashed together to get a hash value for their parents node. Hashing continues until we get to the root node of the B-tree. The hash value of the root node (we will henceforth call it RV, following Micali's notation) plus some additional information, like the date and time, is then signed by CA. We assume collision-free hash function. The Merkle tree can be constructed incrementally (for a new CA) or all at once (for an existing CA) and can be updated incrementally on a regular (e.g., daily) basis. (With 2-3 trees, each certificate update only requires an $O(\log n)$ work for the directory, where n is the total number of certificates.)

Define a *cert-path* for a c-statement to be the path from the leaf containing the c-statement to the root, along with the hash values necessary to verify that path. (This includes the hash values for all siblings of nodes along that path.) When the user requests a certificate, she is supplied additionally with the cert-path of the c-statement, along with the signature on the RV. This information is sufficient to verify that the certificate has been valid as of the latest update.

In the next two subsections, we discuss how to handle CA key change, and historical queries. Our EFEKT system handles both situations much better than all previous systems. Following that, we discuss how the EFEKT system allows for many more efficiency optimizations.

CA Key Change The CA key may need to be changed due to being compromised, or possibly as a simple security measure, e.g., a move from a 2048-bit key to a 4096-bit key to improve security. In any case, our system handles the change in a reasonable way. The CA simply generates a new key pair, broadcasts the new public key using some method for authentication (perhaps publishing in the NY Times in the case of key compromise, or simply signing the new key with the old one in the case of a scheduled update), signs the current root value with the new private key, and uses the new private key for all future signatures.

In the situation of the CA private key being compromised, our system also has the advantage that creating counterfeit certificates is difficult, because for that the forger has to fake the root of the CVT, and this is much easier to notice than a stand-alone bogus certificate. In previous schemes with individually-signed certificates, compromise of the CA's private key forces the CA to revoke all existing certificates, reissue those that are still valid, and somehow supply those new certificates to their owners.

Historical Queries There are two radically different types of certificate usage: ephemeral and persistent. When establishing a connection to a web server we only care if a certificate is valid for the current session. In this case the model described above is adequate. However, if we buy a house, a non-repudiation requirement comes into play. 10 years from now we have to be able to prove that our key was valid on the date of the contract.

To allow historical queries in our system, the CA should store the roots of the CVTs (one root per update period) and the corresponding signatures. For any contract that needs persistent certificates, the cert-paths of the necessary c-statements should be stored with the contract, along with the signature on the root. In case the CA key has not changed since the contract, the historical validity of the c-statements can be verified offline.

Note that in case of a CA key change, the CA should sign all old roots with a new key. Then one can verify historical certificates by checking the current signature on the old root. This is sufficient to maintain historical technical non-repudiation.

Our solution is much more efficient than previous schemes, in which the CA or another party would essentially have to store all certificates ever issued, along with complete CRLs for every update period. In the event of a key change, it would need to resign all certificates ever issued, and all the old CRLs.

Vendor Caching for Efficiency For verifiers that check many signatures every day (like vendors or routers using certificates to establish secure communications) communication with directories can be reduced by caching the top part of the CVT. This top part together with a CA's signature can be verified once a day. Then only the lower part of the path needs to be checked to validate the certificates and only hashes need to be performed.

The savings are most pronounced for OCSP. According to Visa, VisaNet, Visa's processing system, transmits more than 2,700 messages per second dur-

ing peak season. This amounts to over 64 million messages per day. One RSA signature can be computed in 1 second, which means that more than 2700 computers are needed just to verify signatures. Visa claims around 850,000,000 cards currently on the market. This means the B-tree height is at most 30. Caching the top 10 levels we obtain 20 hashes are necessary to validate the signature in EFECT. Using Rivest's figure that hashing is 10,000 faster than signing, we obtain a speedup of 500, which results in using 6 computers instead of 2700.

User Caching for Efficiency A user may reduce the need for online verification checking by storing the current cert-path for the c-statement, say on a smartcard. For instance, if a user's certificate is needed making purchases throughout a day, a user could download the cert-path in the morning and make quite a few purchases during the day without any need to contact a directory. If a vendor has not stored the RV signature for the day, then the signature would also have to be stored on the smart card for totally offline verification.

4 Advantages

Here we outline the major advantageous features of the EFECT system. (Some were discussed in the previous section.)

4.1 Efficiency

CA computation By not having to individually sign certificates, the CA performs much less computation. According to Rivest, hashing is about 10,000 times faster than signing. Since each insertion of a certificate into the Merkle tree requires about 30 new hashes, there is about a factor of 300 speedup.

Vendor computation A vendor only needs to verify one signature per day, that of the root of the CTV.

Communication Complexity Eradicating signatures from certificates significantly reduces CA-directory communication.

Suppose we deal with attribute certificates, that are short (say 100 bits or less), have short life span and are created frequently. If updates of the tree happen every minute and 100 certificates are created on average during this time providing short term access privileges that last from minutes to hours to days, say, to listen to a CD, or watch the movie, then let us see what bandwidth savings a directory will get. In an hour 100 certificates plus 1 signature will consume about 11Kbit. 100 certificates with 100 signatures will consume about 110 K bit, i.e., our scheme will provide about tenfold savings of bandwidth in this case. This means that our scheme will not be hampered even by 28Kbps connection, while any scheme that requires individual signatures will need at least 4 such links even for this simple transfer of information.

Offline shopping One can easily download the current freshness information from a directory, containing the hash values along the path from a leaf to the root and the signature of the root. This is enough information to perform certification off-line for the day, and would thus allow off-line shopping. (A similar method was pointed out in [NaorNissim].)

User storage All verification information necessary is stored in the directory. This might be important in situations where a user has a device with limited available memory, such as a smart card or digital phone. This feature could also be important in preventing “chosen protocol” attacks [KSW] (see Appendix A).

4.2 Security

CA may use more secure keys Since the CA only needs to compute one signature per day, and the vendors only need to verify one signature per day, the CA can use longer (i.e., more secure) keys in its signature computation. Security is also improved since there are fewer signatures available for use in cryptanalysis.

CA may use multiple keys The root signature could be signed by multiple CA keys, stored at different locations, with a public key sharing scheme or using multiple signature schemes. These would only need to be checked if the main CA key is ever compromised. Thus it provides much stronger “backup” protection. If a signature scheme is ever broken, the resulting key compromise will be much easier to repair. With only one signature a day to compute one can apply more computationally intensive signature schemes, like Merkle’s, that are sufficiently different from those currently in use so that they are not likely to be broken by advances in factoring or discrete logarithms solutions. This would be more difficult in other schemes, in which *every* certificate would need to be signed with the backup key.

Protecting CA keys We can additionally protect the CA keys by performing “the signature of the day” using a threshold scheme. Computers that participate in the scheme can be physically positioned in different locations, so that even a burglary would not expose the key. Also, some of those computers can be small and cheap and used solely for the purpose of computing the signatures. They can be offline or even turned off all the time except for the “signature” time, which will make a break in exceedingly difficult. If any of these machines is compromised, the rest can perform proactivisation [HJJKY96,FGMY] and thus make the exposure absolutely useless for the perpetrator.

Less damage from CA key compromise As discussed above, the CA only has to broadcast a new public key and sign all the old roots. In other schemes, the CA would have to resign all old certificates along with the daily certificate revocation structures.

4.3 Miscellaneous

Easy changes To change the information in a c-statement, there does not need to be a revoked certificate, and a new certificate. Instead, the information is simply changed, and the change is noted in the next day's CVT. This does not give any extra power to a CA, because under current rules CA can change data in a certificate compared to a certificate request, and is otherwise more or less omnipotent in regards to the certificates it issues.

Unidirectional communication and prescheduled updates Suppose that you go on a business trip and get a special certificate that allows you to access your computer from the laptop you take on that trip. Since a lot of laptops get stolen, you want to set expiration date in a week. However, it happens that you need to stay longer and thus need to extend the validity period of your certificate. In any scheme that requires a certificate to be signed, you will have to request the changes and to get the signature back. In our scheme it is sufficient to leave a request, say, on the answering machine of a responsible person and have your certificate extended without any need for anybody to get in touch with you. On a long trip you can schedule weekly extensions in advance, so that communication is required only in case of emergency.

Atomic Certificates Atomic certificates had been introduced recently in [atomic] for the purpose of better privacy protection. Instead of creating one “catch-all” certificate, many small pieces are certified by a CA. Then their owner can combine those pieces and create a certificate that will only contain fields corresponding to a particular task at hand. Thus, a certificate for one’s bank will most probably contain the Social Security Number, while a certificate for merchant - won’t. The main problem with this approach in the traditional model is that each “atomic certificate” must be signed by a CA. Our solution eliminates this bottleneck.

One-time Certificates The author of the “Atomic Certificates” draft proposes that after atomic certs are combined into one cert, the owner of that cert signs it with his secret key. We suggest that this idea can be taken one step further by adding date and time to the compendium. This will allow to use them as one-time certificates to prevent replay attacks. For example, if an agent submits queries to a secure database on behalf of its users, and this database checks user’s certificates to verify their authorization to access data, a rogue agent may try to replay a certificate of a user with the top secret clearance, to obtain data for another, untrusted user. One-time certificates can be useful in preventing this sort of attack.

5 Comparison to Previous Systems

5.1 X.509 Certificates and Certificate Revocation Lists (CRL)

Our system has advantages over CRLs in that the full CRL (which could be quite lengthy with a standard revocation rate of 10%) must be sent to the verifier to

assure a certificate is still valid on a certain date. Another disadvantage is that it is difficult to support non-repudiation because expired certificates are removed from the CRL. Finally, a CRL leaks information about all revoked certificates, which may not be acceptable.

In fact, our model can be used in tandem with X.509 certificates. In this case the issuing CA signature on the certificate, which is now mandatory, can be made optional.

5.2 SDSI/SPKI

The SDSI/SPKI model requires certificates to be validated online, while our scheme allows mostly offline verification.

5.3 Certificate Revocation Trees

The 23CRT scheme of Naor and Nissim [NaorNissim] improves upon the CRT scheme of Kocher [Kocher]. In terms of communication. Our scheme is very similar to the 23CRT scheme. As shown in Naor and Nissim [NaorNissim], that scheme is much improved in terms of CA-to-directory communication over previous schemes, and better than all schemes but Micali's CRS1 scheme in terms of communication for individual queries.

Our model provides the full range of services: it allows to find certificates, prevent their forgery, supports archiving and mobile devices with limited resources, such as smartcards and mobile phones.

CRT does none of these things. CRT and 23CRT deal exclusively with revocation. However, in cases when other means are employed to obtain certificates and guarantee their validity, and only revocation information is needed, Naor and Nissim's scheme can be used, since its communication for CA-directory interaction is about 10 times less than in our scheme (assuming a 10 percent revocation rate).

5.4 Other Differences

Except for CRS2, none of the previous systems can prove certificate non-existence. This is because CRLs and CRTs provide no indication at all about existence of non-revoked certificates. In the CRS2 scheme, if the CA key is quietly compromised, a forger can create fake certificates with the same certificate numbers as real certificates and this will be almost impossible to detect. This activity is even assisted by the CRS2's propensity to notify the relying party of the status of neighboring certificates. Only our model makes forging certificates very difficult to hide.

Other new features in our model are unidirectional and prescheduled certificate modification (e.g. validity extension).

Note that only CSR1 and EFECT do not provide any information about the status of unrelated certificates.

6 Conclusion

We have described a technique whereby a company like VISA, endorsing roughly 2^{30} certificates, can provide near-current information to merchants as to which of its customers is certified and whose credit privileges have been revoked in a near-offline manner.

We have demonstrated how computation time, memory requirements, and communication complexity can be cut simultaneously by using a better data structure and better protocols. We do not need to put any additional trust into directories to achieve these dramatic savings, nor do we need to put any additional limitations on any of the parties.

Our approach also significantly reduces data available for cryptanalysis of the CA's keys, allows for better protection of those keys, as well as provides gracious CA key change in cases of the regular update, of key compromise, and even when one of the signature algorithms used is broken.

We introduce additional enhancements such as mostly offline revocation check and uni-directional certificate update.

7 Acknowledgements

The authors would like to thank unknown referees for thoughtful comments which helped to improve the paper.

References

- [Anderson] Ross Anderson *Why Cryptosystems Fail*, Fairfax 93
- [CRL] *Internet X.509 Public Key Infrastructure, Certificate and CRL Profile*, Internet Draft, PKIX Working Group
<http://www.ietf.org/internet-drafts/draft-ietf-pkix-ipki-part1-10.txt>
- [DH] Diffie and Hellman *New Directions in Cryptography*, IEEE Transactions on Information Theory IT-22, 6 (Nov. 1976), 644-654 [346](#)
- [Kohnfelder] Loren Kohnfelder, *Towards a Practical Public-key Cryptosystem*, Bachelor's thesis, MIT, May, 1978 [346](#)
- [EGM] S. Even, O. Goldreich and S. Micali, *On-Line/Off-Line Digital Signatures*, Journal of Cryptology 1996, pp. 35-67. [344](#)
- [FGMY] Y. Frankel, P. Gemmell, P. MacKenzie and M. Yung. *Proactive RSA*, Crypto 97. [349](#)
- [HJJKY96] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, *Proactive public key and signature systems*, The 4-th ACM Symp. on Comp. and Comm. Security. April 1997. [349](#)
- [JY] C. Jutla and M. Yung, *PayTree: "Amortized-Signature" for Flexible MicroPayments* Second Usenix Workshop on Electronic Commerce, 1996. [345](#)
- [KSW] J. Kelsey, B. Schneier, and D. Wagner, *Protocol Interactions and the Chosen Protocol Attack* Advances in Cryptology: CRYPTO '98. [349](#), [353](#)

- [Kocher] Paul Kocher, *A Quick Introduction to Certificate Revocation Trees (CRTs)*
<http://www.valicert.com/technology/> 343, 345, 351
- [Merkle] R. Merkle, *A Certified Digital Signature*, Advances in Cryptology: CRYPTO '89, pp. 218-238. 343
- [Micali] S. Micali, *Efficient Certificate Revocation*, RSA Data Security Conference, San Francisco, California, January, 1997. 343, 344
- [NaorNissim] M. Naor, K. Nissim, *Certificate Revocation and Certificate Update* Proceedings of Usenix'98. 343, 345, 349, 351
- [PKIX] *Internet X.509 Public Key Infrastructure, PKIX Roadmap*, Internet draft, PKIX Working Group
<http://www.ietf.org/internet-drafts/draft-ietf-pkix-roadmap-00.txt>
- [atomic] *Internet X.509 Public Key Infrastructure, ATOMIC CERTIFICATES*, Internet draft, Narayan Raghu, IBM Global Services India ltd.
<http://www.ietf.org/internet-drafts/draft-raghu-atomic-certificates-00.txt> 350
- [Rivest] *Can We Eliminate Revocation Lists?*, Proceedings of Financial Cryptography 1998. A link to this paper can be found at
<http://theory.lcs.mit.edu/~rivest/publications.html> 343
- [SDSI] links to SDSI and SPKI materials can be found at
<http://theory.lcs.mit.edu/~cis/sdsi.html> 345

A Chosen-protocol attacks

The recent result on “chosen protocol” attacks [KSW] shows how dangerous it is to use the same certificate in two apparently unrelated applications. Thus, an owner of a smartcard (or other certificate holding device with a limited memory) may want to put on it as many certificates as can possibly fit. If the available memory is 2K and the desirable key length for CA is 2048 bits, then for every scheme that requires each certificate to be signed at least 2148 bits will be needed for a 100 bit certificates. Then no more than 7 certificates can fit on a card. By removing the signatures, 327 certificates can fit on the same card. For offline shopping, using our scheme with a CVT of height 20 and a hash of length 128 bits, we can fit more than 30 certificates on the same card: more than a 4-fold improvement in the number of certificates (compared to the standard scheme with individual signatures) plus better functionality.

Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions

Ronald Cramer¹, Ivan Damgård², and Philip MacKenzie³

¹ Institute for Theoretical Computer Science

ETH Zurich, 8092 Zurich

cramer@inf.ethz.ch

² Aarhus University, BRICS

³ Information Sciences Research Center, Bell Laboratories

600 Mountain Ave., Murray Hill, NJ 07974-0636

philmac@research.bell-labs.com

Abstract. We initiate the investigation of the class of relations that admit extremely efficient perfect zero knowledge proofs of knowledge: constant number of rounds, communication linear in the length of the statement and the witness, and negligible knowledge error. In its most general incarnation, our result says that for relations that have a particular three-move honest-verifier zero-knowledge (HVZK) proof of knowledge, and which admit a particular three-move HVZK proof of knowledge for an associated commitment relation, perfect zero knowledge (against a general verifier) can be achieved essentially for free, even when proving statements on several instances combined under monotone function composition. In addition, perfect zero-knowledge is achieved with an optimal 4-moves. Instantiations of our main protocol lead to efficient perfect ZK proofs of knowledge of discrete logarithms and RSA-roots, or more generally, q -one-way group homomorphisms. None of our results rely on intractability assumptions.

1 Introduction

1.1 The Problem and the Motivation

Suppose a prover P would like to efficiently convince a verifier V about knowledge of some secret, for instance, a discrete logarithm or an RSA root. More formally, we want to prove knowledge of a secret witness to a public value over some given relation. Of course, a trivial way to do this would be to simply reveal the secret. However, this is useless in a cryptographic context – we want a solution allowing P to keep the secret to himself. More concretely, we want an efficient zero-knowledge proof of knowledge for the given relation.

In this paper we characterize a class of relations where such a zero-knowledge proof of knowledge can be built without making any computational assumptions, and at negligible extra cost (communication and number of moves) compared to the trivial non zero-knowledge solution. Related results in this vein were shown

before in [CD98], but for the case of proofs of language membership (concretely, for the boolean-circuit satisfiability problem).

Before describing our results in more detail, we emphasize that the fact that we use no intractability assumptions implies that soundness and zero-knowledge of our protocols hold independently of the hardness of any problem, *including even the problems for which we build our proofs of knowledge*. Even if this is nice from a theoretical point of view, one could argue that in practice, there is no use in making a zero-knowledge proof of knowledge for a problem that does not satisfy an intractability assumption: if anyone can compute the prover's secret, there is not any sense in trying to hide it.

There are two answers to this objection: first it implicitly assumes that computational problems fall in two classes, where you have on one side problems for which an intractability assumption holds, and on the other side easy problems. This is in fact not the case. Cryptographic assumptions require that the problem be hard on average, whereas problems that are hard in the worst case, but often easy, fall in neither of the two classes. For instance, graph isomorphism seems to be one such case. Thus there could well be cases where one would like a zero-knowledge proof for a problem, even though one might be reluctant to base an intractability assumption on it.

Secondly, it should be noted that a protocol with no assumptions is much easier to use as subprotocol in a larger construction (and this is indeed an important type of application of ZK proofs). Let us elaborate on this: In any multiplayer protocol it is an obvious advantage if we can prove it secure against cheating by player A , even if A is computationally unbounded. Indeed many cryptographic tasks do allow for solutions where some (though not all!) players can be unbounded. However, if in constructing such a solution, we use as subprotocol a ZK proof that only works if one of the players, say A , has bounded computing power, then nothing we build on top will allow us to prove the overall protocol secure against an unbounded A . So we see that using subprotocols with no assumptions allows more design freedom in deciding which players should be protected against unconditionally.

1.2 Our Results

Our results apply to relations admitting a proof of knowledge of a special form: a prover P can convince in perfect zero-knowledge an *honest* verifier using a 3-move protocol, where P sends the first message, V sends a random challenge, and P answers the challenge. We assume that, whereas a truthful P can answer any challenge, a cheating prover can answer at most one. We call this a Σ -protocol (a precise definition follows below). There are several known protocols of this form, e.g., [Sch89] for discrete logs and [GQ88] for RSA roots. These protocols have the efficiency we are after: constant round and communication a constant factor larger than the length of the secret. But they cannot be proved to be zero-knowledge against a dishonest verifier - at least not by any known resettable simulation technique [GK96b].

Our main result is as follows: it was observed in [Dam89,FS89] that any Σ -protocol for a relation R leads to existence of a commitment scheme, which in turn naturally defines a new relation R' (see below for details). Loosely speaking, R' consists of pairs in which one component is a commitment and the other is a string opening that commitment. We consider the case where both R and R' have Σ -protocols. This class includes the cases of discrete log, RSA roots, and in general any relation built from a q -one-way function [CD98].¹ For any such relation, we obtain a 4-move protocol that is perfect zero-knowledge in general *without making any computational assumptions*. Note that this in particular means - somewhat surprisingly - that we do not need to assume that the commitment scheme associate to R is secure. We just use it as a building block in the protocol.

Our results come at the price of one extra move and a small constant factor of communication compared to the Σ -protocols we start from. The 4 moves is optimal for protocols that are black-box zero-knowledge [GK96b]. Additionally, we obtain a 6-move protocol with an even smaller constant factor of communication overhead.²

When instantiated for concrete problems like discrete log and RSA, we get very practical perfect zero-knowledge protocols which naturally have many applications, including improving the efficiency of many distributed cryptographic protocols (e.g., [GR98,FMY98]). As an example of the practicality of these protocols, consider the problem of discrete logs in Z_p^* (p prime) where $|p|$ denotes the bit length of p . We present a 4-round ZK proof of knowledge for this relation that communicates only $9|p|$ bits. Additionally it only requires 10 exponentiations (6 by V , 4 by P), so there is only a small constant factor of computation overhead, also. Our 6-move version of this protocol communicates $8|p|$ bits and requires only 7 exponentiations (4 by V , 3 by P). The knowledge error for both proofs is simply $1/q$, where q is the order of the generator used. As another example, we mention one case where our protocols are more efficient than any previous ones, with or without intractability assumptions, namely the case of proving knowledge of an RSA root where the public exponent is a small prime, say 3. If we try to apply Guillou-Quisquater [GQ88] directly to this situation, we can only get negligible error probability if we iterate their protocol many times because the error probability of one instance is $1/3$ in this case. By contrast, a variant of our construction allows to give a perfect ZK proof of knowledge of an RSA third root with the same asymptotic efficiency as for RSA roots with large public exponent. Specifically, to achieve knowledge error 3^{-t} for an RSA modulus n , our protocol runs in 4 rounds and communicates only $11 \log |n| + 5t$ bits, as compared to $\Omega(t)$ rounds and $\Omega(t \log |n|)$ bits for the iterated GQ protocol.

Our methods generalize to provide perfect zero-knowledge proofs of partial knowledge, allowing P to, for example, prove that he knows at least t out of $n > t$

¹ Actually this class includes random self-reducible languages also (i.e., our result encompasses all the languages considered in [BMO90,SKS91]), but we are mainly concerned with those languages with efficient (linear communication) Σ -protocols.

² Due to space limitations, the 6-move protocol can only be found in the full version.

secrets, in particular without releasing any information about which secrets are known. This may be seen as a generalization of the work by Cramer, Damgård and Schoenmakers [CDS94] who provide efficient witness hiding protocols for the same type of problems. In this context, our work shows that the stronger property of perfect zero-knowledge can be obtained at the cost of one extra move in the protocol. Monotone compositions of ZK were also studied in [SCP93] and [SCPY94], but their protocols are not as efficient as ours, in fact we exhibit example cases, where our protocols are super polynomially more efficient.

1.3 Related Work

There are several earlier results on building general zero-knowledge proofs from honest-verifier zero-knowledge protocols of this type. A well-known technique that achieves constant round protocols (which was only proven secure quite recently [GK96a]) is to let V commit to his challenge before P sends his first message. Apart from the fact that this would require a computational assumption, the method only seems to work for proofs of language membership. Damgård [Dam93] shows a method without computational assumptions, which however leads to a non-constant number of moves. Subsequent results, such as [GSV98], also lead to a non-constant number of moves, and actually do not apply to proofs of knowledge.

Also, there are methods for building constant-round protocols from scratch: Bellare, Micali and Ostrovsky [BMO90] show that 5-move perfect zero-knowledge proofs are possible for any random self-reducible relation.³ Saito, Kurosawa and Sakurai [SKS91] improve this to 4-moves. Feige and Shamir [FS89] show a general method for constructing constant-round ZK proofs of knowledge. Their 4-move protocol relies on the hardness of discrete log, and their 5-move protocol relies on the existence of a one-way function. The protocols work for any NP relation, but do not lead to practical protocols. Bellare, Jakobsson, and Yung [BJY97] construct a four-round (computational) proof of knowledge for any NP-relation that is computationally ZK, and relies only on the existence of a one-way function. Again, this is not a practical protocol. None of these results achieve linear communication.

1.4 Road Map to the Paper

Section 3 presents most of the notation, concepts, definitions, and model we use. Section 4 presents the 4-move perfect ZK proof-of-knowledge protocol. Section 5 presents security proofs for these protocols. Section 6 present some examples of how the general protocol can be instantiated. Section 7 presents an extension to monotone compositions.

³ They prove this for language membership, but it is not hard to show that their protocol is also a proof of knowledge.

2 Definitions of Proofs of Knowledge and Perfect Zero-Knowledge

For a *Proof of Knowledge* (P, V) for some binary relation $R = \{(\alpha, \beta)\}$ we use the definition of Bellare and Goldreich [BG92]. This definition includes the usual *completeness* condition which says that a prover indeed knowing what he claims he knows is accepted by the verifier. For convenience we (informally) state the technical soundness condition.

There is a function $\kappa : \{0, 1\}^* \rightarrow [0, 1]$ (*knowledge error*) and a probabilistic expected polynomial time oracle machine E (*knowledge extractor*) such that the following holds.

Let P be an arbitrary prover (not necessarily following the protocol!), claiming to know a witness β for a given public string α , and let $\epsilon(\alpha)$ denote P 's success probability. Then E , having rewritable black-box access to the prover P , either outputs some witness β for α , or a special halting-symbol. Furthermore, the probability that E outputs a witness is greater than or equal to $\epsilon(\alpha) - \kappa(\alpha)$.

In our context *Perfect Zero-Knowledge* is defined by means of the usual black-box formulation [Gol95], but with perfect indistinguishability of conversations produced by the expected polynomial time universal simulator, having rewritable black-box access to the verifier, and the conversations resulting from “real life” interactions between prover and verifier.

3 Definition and Basic Theory of Σ -Protocols

We overview the basic definitions and properties of our primitive Σ -protocols. Let a Σ -protocol (A, B) be a three move interactive protocol between a probabilistic polynomial-time prover A and a probabilistic polynomial-time verifier B , where the prover acts first. The verifier is only required to send random bits as a challenge to the prover.

More precisely, let $R = \{(\alpha, \beta)\}$ be a binary relation and assume that for some given polynomial $p(\cdot)$ it holds that $|\beta| \leq p(|\alpha|)$ for all $(\alpha, \beta) \in R$. Furthermore, let R be testable in polynomial time, and let R^* denote the collection of strings α such that, for some string β , $(\alpha, \beta) \in R$. The string β is called a witness for α . For some $(\alpha, \beta) \in R$, the common input to both players is α while β is private input to the prover. For such given α , let (a, c, z) denote the conversation between the prover and the verifier. To compute the first and final messages, the prover invokes efficient algorithms $a(\cdot)$ and $z(\cdot)$, respectively, using (α, β) and random bits as input. Using an efficient predicate $\phi(\cdot)$, the verifier decides whether the conversation is accepting with respect to α . The relation R , the algorithms $a(\cdot)$, $z(\cdot)$ and $\phi(\cdot)$ are public. The length of the challenges is denoted t_B , and we assume that t_B only depends on the length of the common string α .

In the present context, we will assume that all Σ -protocols we are given satisfy the following security properties. First, (A, B) satisfies a strong flavour of knowledge soundness: Let (a, c, z) and (a, c', z') be two conversations, that are

accepting for some given α . If $c \neq c'$, then $\alpha \in R^*$ and, on input α and those two conversations, we can efficiently compute β such that $(\alpha, \beta) \in R$. This is called *special soundness*, and the pair of accepting conversations (a, c, z) and (a, c', z') with $c \neq c'$ is called a *collision*.

It can be shown by the results of Damgaard and Pfitzmann [DP] that a Σ -protocol (A, B) with special soundness is a *proof of knowledge* in the sense of Bellare and Goldreich [BG92], with *knowledge error* 2^{-t_B} .⁴

Finally, we assume (A, B) satisfies *special honest verifier zero knowledge* (special HVZK). This means that we are given a (probabilistic polynomial time) simulator M that on input $\alpha \in R^*$ generates accepting conversations with the exactly same distribution as when A and B execute the protocol on common input α (and A is given a witness β for α), and B indeed honestly chooses its challenges uniformly at random. The simulator is special in the sense that it can additionally take a random string c as input, and output an accepting conversation for α where c is the challenge.

A simple but important fact (see [CDS94]) is that if a Σ -protocol is HVZK, the protocol is perfectly *witness indistinguishable* (WI) [FS89]. Although HVZK by itself is defined with respect to a very much restricted verifier, i.e. an honest one, this means that if for a given instance α there are at least two witnesses β , then even an arbitrarily powerful and malicious verifier cannot distinguish which witness the prover uses.

Examples of Σ -protocols can be based on one-way group homomorphisms, claw-free pairs of trapdoor permutations, random self-reducible languages, and q -one-way group homomorphisms [CD98], which are very attractive from an efficiency point of view.

3.1 Partial Proofs

A General Theorem The following material is relevant to Section 7. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$, $f \not\equiv 0, 1$, be a *monotone function*⁵, i.e. for all $x \leq y$ we have $f(x) \leq f(y)$.

The *dual* f^* of f is defined as $f^*(x) = f(x \oplus \mathbf{1}) \oplus 1$, where $\mathbf{1}$ denotes the all-one string. This is a monotone function as well.

There is a natural connection between monotone functions and monotone access structures from secret sharing: Γ_f is the collection of all sets $A \subset \{1, \dots, n\}$ such that $f(A) = 1$, i.e. f applied to the characteristic bit string of A is equal to 1.

Consider an efficient perfect secret sharing scheme \mathcal{S} with access structure Γ_f . Write $\{0, 1\}^t$ for its key-space (the set from which secrets are chosen). It is

⁴ This is a non-trivial observation, but by standard rewinding techniques one can already verify the slightly weaker property that there is a knowledge extractor that runs in expected time polynomial in the running time of the prover and $1/(\epsilon - 2^{-t_B})$, where ϵ is the prover's success probability.

⁵ Usual Boolean ordering: $a \geq b$ iff for all bit positions where a has 1, the corresponding position in b is 1 as well.

efficient in the sense that all operations take time polynomial in n (the number of players) and t . Say that each player receives a number of strings from $\{0, 1\}^t$ as share in a secret. Then the total number of strings dealt to all players is denoted $\text{size}(\mathcal{S})$, the *size* of \mathcal{S} .

We say that \mathcal{S} has *completion*⁶ if the following holds. Let $A \notin \Gamma_f$ be arbitrary and consider the distribution D_A of shares resulting from \mathcal{S} for players in A . By the condition on A and the perfectness of \mathcal{S} , D_A is independent of the secret dealt by the dealer. Completion is satisfied if there exists an efficient probabilistic algorithm that takes as input a random sample from D_A and an arbitrary secret $s \in \{0, 1\}^t$, and outputs a full set of shares consistent with s , with distribution identical to a full set of shares of s generated by the honest dealer in \mathcal{S} .

Let $\mathcal{F} = \{f_n\}_{n>0}$ denote a collection of efficiently computable monotone functions f_n on n bits. Let R be a binary relation as before. For all $f_n \in \mathcal{F}$ and for all l , consider the collection of all tuples $\alpha = (f_n, \alpha_1, \dots, \alpha_n)$ such that $\alpha_i \in \{0, 1\}^l$ for $i = 1 \dots n$. Let $\beta = \{\beta_j\}_{j \in I} \subset \{0, 1\}^*$ be given where $I \subset \{1, \dots, n\}$ and $|\beta_j| \leq p(|\alpha_j|)$ for $j \in I$. Then $(\alpha, \beta) \in R_{\mathcal{F}}$ if and only if $f_n(I) = 1$ and $(\alpha_j, \beta_j) \in R$ for $j \in I$. Note that by our assumptions on R and \mathcal{F} , the composite binary relation can be tested efficiently. The results from [CDS94] imply the following theorem (see also [Cra96] for the full version)

Theorem 1. *Let a Σ -protocol (A, B) for relation R be given, satisfying special honest verifier zero-knowledge and special soundness. Let $\mathcal{F} = \{f_n\}$ be a family of efficiently computable monotone functions. Assume that there is an efficient perfect secret sharing scheme \mathcal{S} with completion for the dual family $\mathcal{F}^* = \{f_n^*\}$ with key-space $\{0, 1\}^{t_B}$.*

Then there exists a Σ -protocol $(\overline{A}, \overline{B})$ for relation $R_{\mathcal{F}}$ satisfying special honest verifier zero-knowledge and special soundness. The size $t_{\overline{B}}$ of the challenges is equal to t_B . The total communication complexity of $(\overline{A}, \overline{B})$ is $\text{size}(\mathcal{S})$ times that of (A, B) plus t_B bits.

Examples of secret sharing schemes satisfying our requirements include *all* efficient linear secret sharing schemes, so in particular Shamir's scheme for the threshold access structure.

A Special Instance: Proof of “OR” In our results to follow, we need a particular, simple instance of the main theorem from [CDS94].

What we use is a slight generalization of a corollary in [CDS94] which enables a prover, given two values (x_1, x_2) , corresponding relations (R_1, R_2) , and corresponding 3-move Σ -protocols $((A_1, B_1), (A_2, B_2))$, to present a 3-move Σ -protocol (A_{or}, B_{or}) for proving knowledge of a w such that either $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$.

We will describe the protocol assuming the challenges from (A_1, B_1) and (A_2, B_2) are of the same length. This can easily be generalized, as long as the challenge length in the combined protocol is at least as long as the challenges from either protocol. The protocol consists of (A_1, B_1) and (A_2, B_2) running in

⁶ This notion corresponds to “semi-smooth” in [CDS94]

parallel, but with the verifier’s challenge c split into $c = c_1 \oplus c_2$ by P , who uses c_1 as the challenge for (A_1, B_1) , and c_2 as the challenge for (A_2, B_2) .

The protocol for A_{or} is as follows: Without loss of generality, say A_{or} knows w such that $(x_1, w) \in R_1$. Let M_2 be the simulator for S_2 . Then A_{or} runs $M_2(x_2)$ to generate (m, e, z) . It sends the first message of (A_1, B_1) , along with m as the first message of (A_2, B_2) . On challenge c , it chooses $c_2 = e$, and $c_1 = c \oplus e$. It is able to provide the final response in (A_1, B_1) because it knows w , and the final response in (A_2, B_2) is simply z .

3.2 Commitments Using Σ -Protocols

For relation $R = \{(x, w)\}$ and Σ -protocol (A, B) , consider the relation $R' = \{((x, a), (c, z))\}$ which holds if (a, c, z) is an accepting conversation for (A, B) on (public) input x . We call R' a commitment relation for (A, B) , since given an input x for which a party A does not know a witness, there is a commitment protocol for A that works as follows:

1. Say A wishes to commit to a value c . A runs the special simulator M for (A, B) with input x and challenge c , producing accepting conversation (a, c, z) for (A, B) .
2. A commits to c by publishing a .
3. A opens the commitment by revealing (c, z) .
4. Anyone can verify this by testing if $\phi(x, a, c, z)$ outputs “accept”

It is easy to see that the commitment is binding as long as A does not know a witness for x .

4 General Protocol

Say a relation R has a Σ -protocol (A, B) with t -bit challenges. Assume the commitment relation R' for (A, B) also has a Σ -protocol (call it (A', B')) satisfying special soundness and special HVZK with t -bit challenges.⁷ Then we show how to construct a 4-move perfect ZK proof of knowledge with knowledge error 2^{-t} for R that does not rely on any intractability assumptions, and whose communication complexity is twice the communication complexity of (A', B') plus the communication complexity of (A, B) plus the number of bits in a commitment. We describe it as a 6-move protocol, but the second and third moves of the first

⁷ Since the special soundness and special HVZK properties are not affected by parallel executions of a protocol, if there exists a Σ -protocol for R' with t' -bit challenges for $t' < t$, then a Σ -protocol with t'' -bit challenges ($t'' = t' \lceil t/t' \rceil \geq t$) can be constructed using parallel executions of the t' -bit challenge protocol. On the other hand, if there exists a Σ -protocol for R' with t' -bit challenges for $t' > t$, then a Σ -protocol with t -bit challenges can be created by simply allowing the prover to choose the remaining $t' - t$ bits of the challenge, and then sending those bits along with z to the verifier. Again, special soundness and SHVZK are preserved.

part can be combined with the first and second moves of the second part, as in [FS89].

In the protocol, say a prover P wishes to prove to V knowledge of a witness w for x , i.e., a w such that $(x, w) \in R$.

Part 1 Using the commitment relation R' and the input x , V commits to value e and proves knowledge of this value using (A', B') . This proof is WI, so does not reveal any information about e . If V does not give an accepting proof, P halts. Otherwise, say the commitment is (x, m) .

Part 2 Let R_{or} be the relation over input pairs $((\alpha_1, \alpha_2), (\beta_1, \beta_2))$ where either $(\alpha_1, \beta_1) \in R$ or $(\alpha_2, \beta_2) \in R'$. Using the Σ -protocol (A_{or}, B_{or}) from Section 6, P gives a WI proof that it knows either a witness for x , or how to open the commitment (x, m) . An honest prover can do this since it knows a witness for x .

Based on the x for which the prover claims to know a witness, the verifier provides a perfectly hiding commitment and a perfect WI proof [FS89] that he can open it. Note that we set it up so that a witness for x can be computed efficiently given any two distinct openings of the commitment.

Next, the prover gives a perfect WI proof of knowledge that he can open the commitment or knows the witness he claims. So why should this convince the verifier that, unconditionally and not relying on intractability assumptions, the prover knows the claimed witness?

This is by the existence of the following knowledge extractor. First, K sets up the commitment and runs, as an honest verifier, the protocol with the prover. Intuitively, by rewinding the prover during the second part of the proof, K either extracts a witness for x , in which case we are done, or extracts an opening of the commitment. In the latter case, and assuming for simplicity that there is an overwhelming number of different ways to open a commitment, we know by the witness indistinguishability of K 's proof of knowledge of an opening that with very high probability the opening extracted from the prover is different from the one known to K . With two different ways to open the commitment, K can find a witness for x and we are done.

The black-box simulation for this protocol intuitively works by rewinding the verifier the simulator extracts an opening for the commitment and thus can give the perfect WI proof to the verifier that he knows the witness or can open the commitment. Thus we achieve perfect zero knowledge.

4.1 A Remark on Computational ZK Proofs of Knowledge

We emphasize that our method allows us to construct perfect ZK proofs of knowledge from Σ -protocols without making any complexity assumptions, *not even on the underlying problem itself*. If we were willing to do that, one could construct a ZK proof of knowledge for any Σ -protocol as follows. Assume there is an invulnerable generator G [FS89] for relation R and assume R has a Σ -protocol. Then given problem instance y , where P claims to know a solution, V

makes instances z_1, z_2 using G and shows that he knows a solution to z_1 or z_2 . This will naturally be witness hiding, since there are at least two witnesses. The P proves she knows a solution to y or z_1 or z_2 . The zero-knowledge property is obtained immediately, and soundness follows by a standard argument, assuming P is poly time and G is invulnerable.

5 Security of the General Protocols

Theorem 2 is proved in the appendix, while the proof of Theorem 3 is omitted due to space limitations.

Theorem 2. *If relation R has a Σ -protocol (A, B) with t -bit challenges, and commitment relation R' associated with (A, B) also has a Σ -protocol (A', B') with t -bit challenges, then R has a 4-move perfect ZK proof of knowledge with no intractability assumptions, with knowledge error at most 2^{-t} , and with communication complexity twice that of (A', B') plus that of (A, B) plus the number of bits in a commitment.*

Theorem 3. *If relation R has a Σ -protocol (A, B) with t -bit challenges, and commitment relation R' associated with (A, B) also has a Σ -protocol (A', B') with t -bit challenges, then R has a 6-move perfect ZK proof of knowledge with no intractability assumptions, with knowledge error at most 2^{-t} , and with communication complexity twice that of (A, B) plus that of (A', B') plus the number of bits in a commitment.*

6 Instantiations of the General Protocol

6.1 q -One-Way Group Homomorphisms (q -OWGH)

Definition q -OWGH's are introduced in [CD98]. We briefly review the definition.⁸

Let q be a fixed prime, and let H and G be finite Abelian groups, with efficient basic operations (as usual: random sampling, equality testing, group operations). Let $f : H \rightarrow G$ be a one-way group homomorphism (easy to compute on elements from its domain, but hard to invert on a random element from its image).

Definition 1. *f is a q -one-way group homomorphism, if given f and an arbitrary x in f 's image, one can efficiently compute $\tilde{w} \in H$ such that $f(\tilde{w}) = x^q$.*

Lemma 1. *Given $f, z, x, 0 < i < q$ with $f(z) = x^i$, one can efficiently compute w with $f(w) = x$.*

To prove the lemma, let $i, j, \beta, z, \tilde{w}$ be such that $ij = 1 + \beta q$, $f(z) = x^i$, $f(\tilde{w}) = x^q$. Then $f(z^j) = x^{1+\beta q}$. Hence, $f(z^j \tilde{w}^{-\beta}) = x$, and define $w = z^j \tilde{w}^{-\beta}$.

⁸ We also remove a redundant requirement from the definition given in [CD98].

Σ -protocols First we give the basic Σ -protocol from [CD98]. A proves knowledge of an f -preimage w of $x \in G$. It starts by A choosing $p \in_R H$ and sending $a = f(p)$ to B . Next, B chooses $c \in_R \{0, \dots, q - 1\}$ and sends c to A . Finally, A sends $z = pw^c$ to B , who checks that $f(z) = ax^c$ (These computations are performed in the group G).

The proof that this basic protocol, which is clearly a generalization of the RSA-protocol from Guillou/Quisquater [GQ88] and the DL-protocol from Schnorr [Sch89], is a Σ -protocol with satisfies completeness, special soundness, and special HVZK is given in Lemma 25 from [CD98]. Note that it is in fact a proof of knowledge with knowledge error $1/q$. (See [CD98] for the proofs that q -OWGH's and the corresponding Σ -protocols can be constructed for RSA and Discrete Logarithms.) It remains to be shown that this Σ -protocol admits a Σ -protocol for the commitment relation.

This means that we have to provide a Σ -protocol for A to prove knowledge of a (e, s) for (x, m) , where (m, e, s) is an accepting conversation for $x \in G$ in the basic Σ -protocol for q -OWGH's. i.e. A has to prove knowledge of e, s such that $f(s)x^{-e} = m$. The Σ -protocol follows:

1. A chooses $\rho \in_R H$ and $\sigma \in_R \{0, \dots, q - 1\}$ and sends $a = f(\rho)x^\sigma$ to B .
2. B chooses $c \in_R \{0, \dots, q - 1\}$ and sends c to A .
3. A has to find $z_1 \in H$ and $z_2 \in \{0, \dots, q - 1\}$ such that $f(z_1)x^{z_2} = am^c$.

A computes \tilde{w} such that $f(\tilde{w}) = x^q$ and integers $0 \leq \alpha < q$ and β such that $\sigma - ec = \alpha + \beta q$. A sends $z_1 = \rho s^c \tilde{w}^\beta$ and $z_2 = \alpha$ to B , who checks that $f(z_1)x^{z_2} = am^c$.

In the general protocol, the \oplus operation for the challenge bits in the “OR” protocol may be replaced by addition mod q .

Assuming that the members of groups G and H can be represented with l bits, the communication complexity of the basic Σ -protocol for q -OWGHs is $3l$, and that of the commitment Σ -protocol for q -OWGHs is $4l$. Therefore, the communication complexity of the 4-move ZK proof of knowledge for q -OWGHs is $12l$, and that of the 6-move ZK proof of knowledge for q -OWGHs is $11l$.

6.2 Optimized Discrete-Log

Figure 1 gives an optimized version of the general protocol for discrete logarithms (DL-1). The proof of knowledge of either how to open the commitment, or a discrete log of X , is actually combined, with B being the first message of both proofs, and z and r taking the place of the challenge split and last messages of both proofs. Its communication complexity is $9|p|$ and it has a total of 10 exponentiations.

6.3 RSA with Prime Exponents

By using the protocol for q -one-way group homomorphisms, we can directly implement a ZK proof of knowledge of RSA decryption when the public exponent

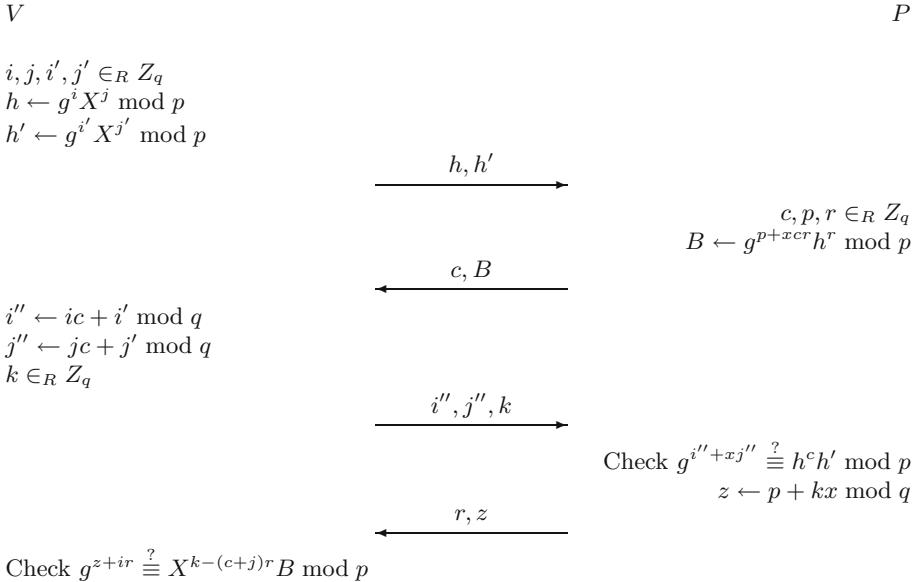


Fig. 1. DL-1: Given a public value X , P proves that it knows the discrete log x of X .

is a prime q . However, the soundness error is $1/q$, which implies that for small q , the protocol might need to be repeated a number of times to achieve negligible soundness error. However, we can also get an efficient 4-move protocol for any small prime RSA exponent q with soundness error q^{-t} assuming that the prover knows a q^t th root of the encryption. (which, for instance, the prover could find if it knew the the factorization of the RSA modulus n). For concreteness, let us assume the public exponent is 3, although the protocol will work for any prime public exponent.

First we give a 3-move Σ -protocol with two parameters, t and t' , which has soundness error 3^{-t} and proves knowledge of a $3^{t'+1}$ th root. It requires that A knows a $3^{t+t'}$ th root. Assume the public input is $y \in Z_n^*$, and that A knows x , where $x^{3^{t+t'}} \equiv y \bmod n$.

Protocol:

1. A chooses $r \in_R Z_n^*$ and sends $m \equiv r^{3^{t+t'}} \bmod n$ to B .
2. B chooses $e \in Z_{3^t}$ and sends e to A .
3. A sends $z \equiv rx^e \bmod n$, and B checks that $z^{3^{t+t'}} = my^e \bmod n$.

In a way, A is attempting to prove that he knows a $3^{t+t'}$ th root, but ends up proving that he knows a $3^{t'+1}$ th root.

Lemma 2. *The protocol above is a Σ -protocol.*

Proof. Completeness is trivial. To prove special soundness, consider two accepting conversations (m, e, z) and (m, e', z') with $e \neq e'$. These give an equation $(z/z')^{3^{t+t'}} = y^{e-e'} \bmod n$. Also, $\gcd(e - e', 3^{t+t'})$ is 3^s for some $s < t$, so by the Extended Euclidean Algorithm, i and j can be found such that $i(e - e') + j3^{t+t'} = 3^s$. Then the equation can be transformed into $((z/z')^i y^j)^{3^{t+t'}} \equiv y^{3^s} \bmod n$. Since raising to 3 is a permutation in Z_n^* , assuming 3 is a valid RSA public key for n , we immediately get a $3^{t'+1}$ th root of y from this. Special HVZK can be shown by using the simulator that on input c chooses r randomly and outputs $(r/y^c, c, r)$.

We do not know how to construct a general commitment protocol for the above Σ -protocol. However, we will be able to substitute a different commitment protocol that works just as well for constructing an efficient 4-move ZK proof for cube roots. We describe the new ZK proof here.

Part 1 First V commits to a value b with commitment $C = y^b r^{3^{2t}} \bmod n$ and proves knowledge of a 3^t th root of C or Cy^{-1} , using the Σ -protocol above with $t' = t$, combined with the Proof of “OR” method of Section 6. This proof is WI, so does not reveal any information about b . If V does not give an accepting proof, P halts.

Part 2 Using Section 6, P proves knowledge of either (1) a cube root of y , (2) a cube root of C , or (3) a cube root of Cy^{-1} . An honest prover can do this since it knows a cube root of y .

As in Section 4, the protocol above can be made into a 4-round protocol by combining the second and third steps of the Σ -protocol of Part 1 with the first and second steps, respectively, of the Σ -protocol of Part 2.

Theorem 4. *The protocol above is a 4-move perfect ZK proof of knowledge of a cube root of y with no intractability assumptions, with knowledge error at most 3^{-t} , and with communication complexity five times that of the Σ -protocol for cube roots (i.e., the Σ -protocol above with $t' = 0$), plus the number of bits in a commitment.*

The proof is similar to that of Theorem 2, and is omitted due to space limitations.

We stress again that the protocol above generalizes to any small prime RSA public key. Also note that we can construct an efficient 6-move protocol for the same problem, as in Section 4.

7 Monotone Composition

We discuss a further application of our results. It's now possible to combine Theorems 2 and Theorem 1:

Theorem 5. *Let relation R have a Σ -protocol (A, B) with t -bit challenges, and let the commitment relation R' associated with (A, B) also have a Σ -protocol*

(A', B') with t -bit challenges. Let $\mathcal{F} = \{f_n\}$ be a family of efficiently computable monotone functions. Assume that there is an efficient perfect secret sharing scheme \mathcal{S} with completion for the dual family $\mathcal{F}^* = \{f_n^*\}$ with key-space $\{0, 1\}^t$. Then relation $R_{\mathcal{F}}$ has a 4-move perfect ZK proof of knowledge with no intractability assumptions, with knowledge error at most 2^{-t} , and with communication complexity $\text{size}(\mathcal{S})$ times that of (A', B') plus $\text{size}(\mathcal{S})$ times that of (A, B) plus the number of bits in a commitment.

To prove this theorem it suffices to show that the protocol $(\overline{A}, \overline{B})$ for relation $R_{\mathcal{F}}$ guaranteed by 1 admits a Σ -protocol for its associated commitment relation $R'_{\mathcal{F}}$. This can be constructed from the Σ -protocol for the commitment relation R' of R , by invoking Theorem 1 on the commitment relation R' and its associated Σ -protocol (A', B') . A similar combination is possible with Theorem 3.

We now identify an interesting class of secret sharing schemes suitable for our purposes, namely *linear secret sharing schemes* (LSSS), and establish an example of a super-polynomial gap with earlier work.

Let K denote a finite field. In an LSSS each player receives some linear combination of the secret $s \in K$ and some random values $\rho \in K$ chosen by the dealer. The distribution of a secret in fact consists of the dealer selecting a random vector with s in its first coordinate and multiplying a public matrix with this vector to get the shares. In the public matrix, each row is associated with a player and a player may have several rows. In Shamir's scheme, for instance, the matrix is a Van der Monde-matrix.

The matrix has the property that the K -span of rows corresponding to players in a set A contains the vector $(1, 0, \dots, 0)$ iff $A \in \Gamma_f$. It is in this sense that Karchmer and Wigderson [KW93] say that the matrix (with its assignments of players to rows), which they call *monotone span program* (MSP), computes the function f . The *size* of an MSP is the number of rows in the matrix, which is equal to the total number of field elements given to players in the corresponding LSSS.

It is easy to see an MSP \mathcal{M} gives rise to efficient perfect secret sharing schemes with completion (the latter comes down to solving a system of linear equations and sampling a random element from its solution space). The time required for all operations is polynomial in $\log(|K|)$ and $\text{size}(\mathcal{M})$.

In the following we set $K = GF(2)$ for simplicity. It's easy to see that to increase the key-space from $\{0, 1\}$ to $\{0, 1\}^t$ it suffices to execute the basic LSSS t times, and it will still be efficient with completion.

It is well known that if f is computed by an MSP of size m then f 's dual f^* is also computed by an MSP of size m . Another relevant fact about MSP's is that all monotone functions f can be computed, and that monotone formula complexity of f is an upperbound on the minimal size of an MSP computing f . Furthermore, there are monotone functions which are computed by polynomial size MSP's (over $GF(2)$), but require super-polynomial monotone *circuits* (the ODDFACTOR function [BGK⁺96]) (and hence its dual ODDFACTOR* also has super-polynomial monotone circuit complexity).

With this in mind, we can now show that our protocol gives in some cases a superpolynomial efficiency improvement over earlier work, namely [SCPY94] on perfect ZK for monotone formula closure of Random Self-reducible Languages. To see this, let the family \mathcal{F} of monotone functions occurring in Theorem 5 be the family of ODDFACTOR functions, and choose as the relation R anything derived from a random self-reducible problem, for instance the Discrete Logarithm problem. Theorem 5 now provides an efficient protocol for the composite language derived from R and \mathcal{F} , while [SCPY94] will have super-polynomial complexity, since that result is polynomial in the monotone formula complexity of the monotone composition function, ODDFACTOR in our example.

References

- BG92. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—CRYPTO ’92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer-Verlag, 1993, 16–20 August 1992. [358](#), [359](#), [370](#), [370](#), [370](#)
- BGK⁺96. László Babai, Anna Gál, János Kollár, Lajos Rónyai, Tibor Szabó, and Avi Wigderson. Extremal bipartite graphs and superpolynomial lower bounds for monotone span programs. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 603–611, Philadelphia, Pennsylvania, 22–24 May 1996. [367](#)
- BJY97. M. Bellare, M. Jakobsson, and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Advances in Cryptology—EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 280–305. Springer-Verlag, 1997. [357](#)
- BMO90. Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 482–493, Baltimore, Maryland, 14–16 May 1990. [356](#), [357](#)
- CD98. R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO ’98*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer-Verlag, 17–21 August 1998. [355](#), [356](#), [359](#), [363](#), [363](#), [364](#), [364](#), [364](#), [372](#)
- CDS94. Ronald Cramer, Ivan Damgård, and Berry 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. [357](#), [359](#), [360](#), [360](#), [360](#), [372](#)
- Cra96. R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI & Univ. of Amsterdam, November 1996. [360](#)
- CRY89. *Advances in Cryptology—CRYPTO ’89*, volume 435 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990, 20–24 August 1989. [368](#), [369](#), [369](#)
- CRY93. *Advances in Cryptology—CRYPTO ’93*, volume 773 of *Lecture Notes in Computer Science*. Springer-Verlag, 22–26 August 1993. [369](#), [369](#)
- Dam89. Ivan Bjerre Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In CRYPTO’89 [CRY89], pages 17–27. [356](#)

- Dam93. Ivan B. Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions (extended abstract). In CRYPTO'93 [CRY93], pages 100–109. 357
- DP. I. Damgård and B. Pfitzmann. Sequential iteration of interactive arguments. journal version of ICALP'98 paper. 359, 371, 372
- FMY98. Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed rsa-key generation. In STOC'98 [STO98], pages 663–672. 356
- FS89. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In CRYPTO'89 [CRY89], pages 526–545. 356, 357, 359, 362, 362, 362
- GK96a. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *Journal of Cryptology*, 9(3):167–189, Summer 1996. 357
- GK96b. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996. 355, 356
- Gol95. Oded Goldreich. Foundations of cryptography (fragments of a book), February 1995. 358
- GQ88. Louis Claude Guillou and Jean-Jacques Quisquater. A “paradoxical” indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 1990, 21–25 August 1988. 355, 356, 364
- GRR98. R. Genarro, M. O. Rabin, and Tal Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages –, 1998. 356
- GSV98. O. Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In STOC'98 [STO98], pages 399–408. 357
- KW93. M. Karchmer and A. Wigderson. Characterizing non-deterministic circuit size. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 532–540, San Diego, California, 16–18 May 1993. 367
- Sch89. C. P. Schnorr. Efficient identification and signatures for smart cards. In CRYPTO'89 [CRY89], pages 239–252. 355, 364
- SCP93. A. De Santis, G. Di Crescenzo, and G. Persiano. Secret sharing and perfect zero knowledge. In CRYPTO'93 [CRY93], pages 73–84. 357
- SCPY94. Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *35th Annual Symposium on Foundations of Computer Science*, pages 454–465, Santa Fe, New Mexico, 20–22 November 1994. IEEE. 357, 368, 368
- SKS91. Takeshi Saito, Kaoru Kurosawa, and Kouichi Sakurai. 4 Move perfect ZKIP of knowledge with no assumption. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 321–330, Fujiyoshida, Japan, 11–14 November 1991. Springer-Verlag. Published 1993. 356, 357
- STO98. *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, Dallas, Texas, 23–26 May 1998. 369, 369

Appendix A: Formal Definitions

Proofs of knowledge were defined in [BG92]. Much of the formalism below is from that paper.

Let R be a relation as defined in Section 3. Let $R(x) = \{y : (x, y) \in R\}$ be the *witness set* of x . Let $L_R = \{x : \exists y \text{ such that } (x, y) \in R\}$.

Let A, B be interactive functions (as defined in [BG92]). Let $A(x)$ denote the interactive function with input x . Let $A(x; B)$ be a random variable describing the output of A when interacting with B on common input x . Let $A_z(x; B)$ be a random variable describing the output of A with auxiliary input z when interacting with B on common input x . Let $\text{tr}_{A,B}(x)$ denote a transcript of the interaction between A and B on common input x . For a verifier V , modeled as an interactive function, let $\text{acc}_V(x)$ be the set of accepting transcripts for V on input x . For a probabilistic oracle machine C , let $C^{A(x)}(x)$ denote a random variable describing the output of C with oracle $A(x)$ and input x , with the probability being over the random choices of M and A . (Technically, we can assume A 's random bits are fixed [BG92].) We can assume that C can query the oracle for any part of a conversation, by supplying the appropriate prefix to the conversation. (In other words, C can arbitrarily “rewind” the interactive function A .)

We say that a function $f(n)$ is *negligible* if for any $\text{poly}(\cdot)$, there is an n' such that for $n > n'$, $f(n) < 1/\text{poly}(n)$.

Definition 2. Let $\kappa : \{0, 1\}^* \rightarrow [0, 1]$. An *interactive proof of knowledge system for a relation R* is a pair of interactive functions (V, P) , where V is computable in probabilistic polynomial-time, satisfying:

1. *Completeness:* $\forall x \in L_R, \Pr(\text{tr}_{V,P}(x) \in \text{acc}_V(x)) = 1$
2. *Soundness:* There exists a probabilistic expected polynomial-time oracle machine E (the knowledge extractor) such that $\forall P' \forall x \in L_R$, it is the case that $E^{P'(x)}(x) \in R(x) \cup \perp$, and $\Pr(E^{P'(x)}(x) \in R(x)) \geq \Pr(\text{tr}_{V,P'}(x) \in \text{acc}_V(x)) - \kappa(x)$.

The \perp symbol is the output of the oracle that indicates “failure.”

All knowledge extractors that we consider satisfy the following slightly stronger soundness property, which basically states that the knowledge extractor can differentiate the prover's failure from its own failure.

- There exists a probabilistic expected polynomial-time oracle machine E such that $\forall P' \forall x$, it is the case that $E^{P'(x)}(x) \in R(x) \cup \perp \cup \Delta$, and
 - $\Pr(E^{P'(x)}(x) = \Delta) = \Pr(\text{tr}_{V,P'}(x) \notin \text{acc}_V(x))$,
 - $\Pr(E^{P'(x)}(x) \in R(x)) \geq \Pr(\text{tr}_{V,P'}(x) \in \text{acc}_V(x)) - \nu(|x|)$, and
 - $\Pr(E^{P'(x)}(x) = \perp) \leq \nu(|x|)$.

In this case, Δ indicates that V would not accept the proof of P' , and \perp indicates that E failed, even though V accepted the proof of P' .

Two random variables S and S' are perfectly indistinguishable if the distributions of S and S' are identical.

Definition 3. A proof system (V, P) is perfect zero knowledge over R if there exists a probabilistic expected polynomial-time oracle machine M (the simulator) such that for any probabilistic polynomial-time V' , for any $(x, y) \in R$, and any auxiliary input z to V' , the two random variables $V'_z(x; P)$ and $M^{V'_z(x)}(x, z)$ are perfectly indistinguishable.

Taking away z from the inputs to M implies that M is a black-box simulator. All simulators in this paper are black-box.

Definition 4. The proof system consisting of a pair of interactive functions (V, P) , where both V and P are computed in probabilistic polynomial time, is witness indistinguishable (WI) over R if for any probabilistic polynomial-time V' , any large enough input x , any $y_1, y_2 \in R(x)$, and for any auxiliary input z to V' , $\text{tr}_{V_z, P_{y_1}}(x)$ and $\text{tr}_{V_z, P_{y_2}}(x)$ are perfectly indistinguishable.

Appendix B: Proofs

Proof of Theorem 2

Proof. **Communication Complexity:** Straightforward.

Completeness: Straightforward.

Soundness: Let E_{or} be the extractor guaranteed by the special soundness property of (A_{or}, B_{or}) [DP]. We construct an extractor E as follows. Say the prover is P^* , and assume its bits are fixed.

1. E performs Part 1 like a true verifier, say using commitment pair $((x, m), (e, s))$. If P^* does not give a well-formed challenge, E outputs Δ .
2. E runs E_{or} using oracle P^* (in its current state after Part 1, i.e., after receiving commitment m , and the proof transcript T from Part 1).
3. Based on the output of E_{or} , E performs different actions:
 - (a) if E_{or} outputs Δ , E outputs Δ and stops. (In this case, P^* failed to provide a valid proof.)
 - (b) If E_{or} outputs \perp , E outputs \perp and stops. (In this case P^* provided a valid proof, but E_{or} failed.)
 - (c) If E_{or} outputs a witness w for x , E outputs w and stops.
 - (d) If E_{or} outputs a witness (e', s') to commitment (x, m) , with $e \neq e'$, E uses the two conversations (m, e, s) and (m, e', s') and the special soundness feature of (A, B) to generate a witness w for x . Then E outputs w and stops.
 - (e) Otherwise E repeatedly performs steps 1 and 2 (sequentially) until either cases 3c or 3d would occur, at which point E finishes as indicated for that case. In step 1, instead of stopping if P^* does not provide a well-formed challenge, E simply rewinds and tries again.

It is obvious that $\Pr(E^{P^*(x)}(x) = \Delta) = \Pr(\text{tr}_{V, P^*}(x) \notin \text{acc}_V(x))$, and that $\Pr(E^{P^*(x)}(x) = \perp)$ (i.e., the soundness error of E) is bounded by the probability

that E_{or} outputs \perp , which by the special soundness of (A_{or}, B_{or}) , is at most 2^{-t} [DP].

To analyze the expected running time of E we use the following fact: given that a witness (e', s') for commitment (x, m) is extracted, the probability that $e = e'$ is 2^{-t} , since the protocol in Part 1 is witness indistinguishable [CDS94] and e is drawn randomly from a set of size 2^t . (We actually only use that the probability that $e = e'$ is at most $1/2$.)

Let q be the probability of E hitting case 3c, 3d, or 3e. Note that this probability is computed for E starting at step 1, and thus is averaged over all possible transcripts produced in step 1, including the possibility of E stopping. The probability of hitting case 3e is at most $q/2$, by the fact stated above, and therefore the probability of either case 3c or 3d occurring (implying success for E) is at least $q/2$. Thus the expected time of E is easily seen to be polynomial.

Zero-Knowledge: We construct a simulator SIM (that uses V^* as a black-box) as follows.

1. SIM plays the part of the P in Part 1 (i.e., the part of B' in (A', B')). If P does not accept the proof (i.e. V^* failed in its proof), SIM halts, like P would.
2. Otherwise SIM repeatedly runs the extractor E' for protocol (A', B') in Part 1 until E' does not output Δ .
3. If E' outputs a witness to the commitment, SIM plays the part of the prover in the (A_{or}, B_{or}) in Part 2, which it can do since it has found a witness to the commitment.
4. If E' outputs \perp , SIM uses the simulator M_{or} to generate a conversation with a random challenge, and attempts to play the prover in the (A_{or}, B_{or}) in Part 2.
5. E' performs one of the following, depending on the challenge of V^* .
 - (a) If V^* does not send a valid challenge, SIM stops, like a true prover would.
 - (b) If V^* actually sends the challenge generated in the simulated conversation, SIM finishes the proof with the simulated responses.
 - (c) If neither of the previous cases hold, then SIM rewinds V^* and generates conversations with random challenges until V^* actually sends the challenge generated in the simulated conversation, so that SIM can finish the proof.

Obviously SIM produces the same distribution as an honest prover (including when V^* misbehaves).

Say the probability of P accepting the proof of V^* in Part 1 is q' . Then the probability SIM reaches step 2 is q' , and the extractor E' produces Δ with probability $1 - q'$. Thus, the overall expected number of rewinds in step 2 is constant.

Since (A', B') has special soundness, we can assume E' works with soundness error at most 2^{-t} [CD98]. Then the probability that the E' outputs \perp (and thus SIM reaches step 4) is at most 2^{-t} . If E' does not output \perp , SIM can finish the

protocol directly. Otherwise, SIM attempts to finish the protocol by guessing the challenge of V^* .

Let q be the probability that V^* sends a valid challenge, given that step 4 is reached. (Note that q will depend on the conversation transcript from step 1.) The total probability of reaching step 5c is at most $q2^{-t}$ (considering the probability of reaching step 4 is at most 2^{-t}). Moreover, the probability of V^* sending a valid challenge in (A_{or}, B_{or}) and SIM guessing it correctly is $q2^{-t}$. So the expected number of rewinds for SIM in step 5c is constant. Thus the total expected running time of SIM is polynomial.

Cryptographic Approaches to Privacy in Forensic DNA Databases

Philip Bohannon¹, Markus Jakobsson¹, and Sukamol Srikwan²

¹ Bell Laboratories, Murray Hill NJ 07974, USA,

{bohannon,markusj}@research.bell-labs.com

² Chulalongkorn University, Bangkok 10330, Thailand

sukamol@mail.sc.chula.ac.th

Abstract. Advances in DNA sequencing technology and human genetics are leading to the availability of inexpensive genetic tests, notably tests for individual predisposition to certain diseases. While such information is often valuable, its availability has raised serious concerns over the privacy of genetic information. These concerns are further heightened when genetic information is gathered into databases. We study access control for one class of such databases, forensic DNA databases, used to match unknown perpetrators against groups of potential suspects – usually convicted criminals. Our key observation is that for legitimate forensic queries, the sensitive information belonging to the target individual is already available to the querying agent in the form of a blood or tissue sample from a crime scene. We show how forensic DNA databases may be implemented so that only legitimate queries are feasible. In particular, a person with unlimited access to the database will be unable to extract information about any individual unless the necessary genetic information for that individual is already known. We develop a general solution framework, and show how to implement databases which handle certain cases of missing or incorrect DNA tests. Our framework and techniques are applicable to the general problem of encrypting information based on partially known or partially correct keys, and its security is based on standard cryptographic assumptions.

1 Introduction

It is becoming increasingly common that law enforcement agencies collect genetic data from individuals who have been convicted of felonies or certain misdemeanors [12,14,21]. This data is being organized into regional databases, and in the United States, the FBI Laboratory’s **C**ombined **D**N**A** **I**n**D**e**S**ystem (CODIS) [26] allows pooling of locally collected information to support nationwide searches. Upon collection of biological materials from the scene of a crime, such a database can be used to identify an individual from the database with almost complete certainty, a so-called “cold hit.” This technique is particularly useful with violent and/or sexual assaults by an unknown perpetrator, and over 200 such “hits” have been made to date [13]. However, there is significant concern by civil rights and privacy groups that collection of DNA data into databases

can lead to violations of privacy [8,20,30,32]. A particular concern is that the existence of the database may make it possible to discriminate *genetically* against individuals appearing in the database. For example, an insurance company with access to a database of genetic information might decline to insure an individual who is at greater risk for a certain disease [2,20,31]. While expansion of DNA databases is valuable from a law-enforcement perspective, the privacy concern would be aggravated if DNA samples were demanded from a wider population, e.g., everybody who applies for a visa or drivers license, all individuals at birth, etc.¹

Our goal is to design forensic DNA databases which are infeasible to use, even by the owners of the database, for violating genetic privacy but remain relatively easy to use for legitimate law-enforcement purposes. For example, it should be infeasible for an attacker to determine information about a person's genetic makeup from information such as name or social security number. Similarly it should be infeasible for an attacker to create a list of persons who exhibit a certain genotype for a certain test. By contrast, the typical *legitimate* use of a forensic DNA database is to determine the identity of an unknown person who has left biological evidence at the scene of a crime, and this should remain an effective operation to perform. We formalize this problem below.

In fact, protection of private information in a database is a common cryptographic problem, and is normally solved by encryption methods and/or by sharing the information between some set of servers, such that a threshold number of them have to cooperate in order to release the information. As an example of the latter technique, the CODIS system identifies individuals in the centralized database only by the lab where their sample was processed and a lab-specified identifier of that sample. Thus, collusion with the labs would be required to associate a profile with a particular individual [37]. However, long-term trust in government agencies to act as guards of citizen's private information is inherently problematic, and the expansion of the current database has raised a number of concerns [13,29]. By contrast, we propose the use of encryption techniques to ensure that only legitimate queries are feasible. Thus, the database would remain secure even in a worst-case scenario in which it enters the public domain, along with user passwords, etc. In this approach, access to the database is controlled by treating the results of a set of DNA tests from an individual as the *key* used to decrypt information revealing the individual's identity. (DNA tests, discussed in the next section, can be thought of as revealing a few bits of information about an individual's genetic makeup.) What makes this approach particularly appealing for use with databases of forensic DNA data is that the information used as a key is exactly the information for which privacy is a concern. Consequently, the only agents who are allowed access to an individual's record are agents who already possess detailed genetic information about that

¹ For example, police in Britain have collected samples voluntarily from all male residents in a village to successfully reduce the suspect pool in a highly publicized crime [8]. A similar approach recently ended in the resolution of a German murder case [17].

individual. For example, the agent may possess a blood or tissue sample from the individual. If the agent in possession of such a sample is an attacker attempting to violate the individual's genetic privacy, searching the database is not helpful since the biological sample may be tested directly to much greater effect. It is in this sense that we claim that genetic information cannot be leaked from our database.

The problem space we consider is characterized by the number and cost of tests available, the accuracy of those tests, and the variation in tests used for forensic purposes in different locales. Since the science of genetic testing is progressing rapidly, we do not make fixed assumptions about these issues. Instead, we present a framework database implementation, and extend that framework with three different "plug-ins" to produce three concrete schemes. The first of these schemes, the *fixed-query solution*, is applicable when individual tests are exceptionally accurate, and the set of tests to be used for forensic searches is globally standardized. This scheme is simple, using a hash function modeled as a random oracle [3] to create an encryption key from an individual's test results. Our second scheme, *fixed-query with error correction*, addresses the issue of test errors. While in this domain false positives can be corrected with additional tests once a suspect is identified, false negatives have a very high cost, possibly allowing a violent criminal to remain uncaught. This scheme allows correction of a small number of testing errors by adapting recent *fuzzy commitments* techniques developed for use with biometric passwords [15]. Our third scheme, the *flexible-query solution*, addresses the situation where different, overlapping sets of tests may be available. Thus, the set of tests used to enter one individual may differ from the set used for another individual, neither of which may exactly match the set of tests available for analyzing a particular crime-scene sample. (However, in all cases, the set of tests used on a sample from a crime scene must significantly overlap the tests used on individuals in the database.) The use of differing sets of tests may arise simply due to continued evolution of testing technology, or due to differing standards in different countries. This solution is based on secret sharing [35], and also allows limited correction of errors if extra tests results for a crime-scene sample are available.

For such schemes to be employed in practice, sufficiently many tests must be run on each individual so that a key constructed from these tests can be resistant to exhaustive search. Modern and easily automated tests (see Sect. 2) seem to produce around 5.2-6 bits of entropy per test. Currently, the CODIS system specifies 13 tests [5], leading to up to 67-78 bits of entropy, which is more or less sufficient for our schemes. However, it is not clear if all tests are being used in all states, or if all tests produce 5 bits of entropy. Due to dramatic, on-going improvements in automatic testing technology (see, e.g. [34]) as well as the existence of thousands of potential locations on the human genome for new tests,² we assume in this paper that the requisite number of tests for our security parameters will be widely available in the reasonably near future.

² These locations have been identified as a by-product of the Human Genome Project [9,34].

The organization of the paper is as follows: an overview of forensic DNA databases follows in Sect. 2, followed by a discussion of related work in Sect. 3. We introduce the formal model of the problem space and our correctness requirements in Sect. 4. In Sect. 5, we present schemes to address several points in the problem space. Security is argued in Sect. 6. We present our conclusions and discuss various open problems and future work in Sect. 7.

2 Overview of Forensic DNA Techniques and Databases

In this section we introduce terminology and give a short overview of forensic DNA testing and data banking technology. More complete introductions are widely available (see, for example, [16,24]). A *DNA profile* is the combination of DNA test results from a particular individual. A *DNA test* reveals information about the DNA sequence present at a particular location, or *locus*, on the genome of that individual. The possible values present at a locus are referred to as *alleles*. For forensic tests, the alleles are usually expressed as the number of times a known DNA subsequence repeats at a given locus. Since human DNA consists of paired chromosomes, the term *genotype* is used to refer to the combination of alleles present at a particular locus on the two paired chromosomes. For privacy, an important attribute of a locus is the role played by the region of which the locus is a part in the biological function of an individual. In particular, regions of DNA which *code* for particular proteins are referred to as *genes*, and other regions are referred to as *non-coding regions*. In summary, a DNA test result reveals the genotype at a locus in either a coding or non-coding region, and a DNA profile is a set of such test results for an individual.

Human DNA profiles may be matched for a wide variety of reasons, including paternity tests and for identifying remains in wartime, as well as for forensic purposes. As described in the introduction, forensic DNA profiles are used primarily to match blood or tissue sample from a crime scene with that of a suspect from the database. (Other forensic uses include connecting unsolved crimes and identifying victims.) When such a match is used to prosecute a crime, statistical methods and databases of allele frequencies are used to obtain conservative estimates of the probability that the match is significant. A match is *significant* if it is due to the sample originating with the suspect rather than the random event that the profile for the suspect matches that of an unknown donor. For some sites, allele frequencies vary significantly with different ethnic populations, so the above estimates must take into account the reference population from which the forensic sample was derived (see [24]). In fact, the significance of a match on a set of tests is related to the security of an encryption scheme based on those tests – the latter issue is discussed further in Sect. 4.3.

Clearly, a locus with a wide variety of possible alleles is preferable to ensure that significant matches can be obtained with a minimum number of tests. When forensic DNA tests became available in the early 1980s, the technology used was relatively slow and expensive, and a significant amount of the blood or tissue sample left at a crime scene was consumed by each test [24]. While

a single test could distinguish a relatively large number of alleles (up to a few hundred), results were approximate, further complicating the calculations described above. With advances in molecular biology, automated procedures allow DNA to be extracted and “amplified” from a minute quantity of blood or other tissue (for example from saliva on a cigarette butt), allowing tests to be repeated at different labs to improve confidence. Furthermore, the results of these tests are discrete and estimation is not required. However, each test produces fewer alleles, requiring more tests to ensure significant matches. The evolution of testing technology has also had implications for the privacy risk of forensic data. While older tests were drawn exclusively from non-coding regions, modern tests are sometimes from genes [20,24]. Further, regions similar in structure to those used for testing have been determined to be correlated with certain diseases [18,36,38]. While loci currently used for forensic DNA testing are not known to be correlated with diseases, there is no guarantee that further study will not lead to the discovery of just such a correlation. Thus, any substantial forensic DNA database represents the risk that it may be possible in the future to use the database to correlate individuals with a genetic disease or some other biological trait. When designing the database, we therefore make the pessimistic assumption that *all* recordable information about an individual’s DNA sequence should be kept private.

As mentioned in the introduction, several countries are creating forensic genetic databanks [6,28], and almost all states across the USA are creating DNA databanks using the FBI’s CODIS system [26]. CODIS stores information in two indexes: 1) The Convicted Offender Index, which contains DNA profile of individual convicted of crimes, and 2) the Forensic Index, containing DNA profiles developed from crime scene specimens. Each index stores a specimen identifier, the laboratory identifier, names of laboratory personnel in charge of the DNA profile, and the DNA profiles. Information such as case-related information and social security numbers are not included in CODIS. When CODIS identifies a potential match, the laboratories responsible for the matching profiles contact each other to validate or refute the match. After a match has been confirmed by qualified DNA analysts, laboratories may exchange additional information including the identity and location of the identified suspect. More than forty states have passed legislation that requires convicted offenders to provide samples for DNA database. These states have collected approximately 450,000 DNA samples, and analyzed over 150,000. However, a review of DNA data-banking laws observed that little attention is paid to issues of genetic privacy in most of the state legislation [21].

3 Related Work

We consider how to safeguard a database against abuse. Here the somewhat imprecise term “abuse” corresponds to reading private information about individuals in the database when some “key” information has not already been gathered, in this case as evidence, by the querying agent. Our concept allows

the holder of the database to obtain the answer to the query after inputting some record-specific and identifying pieces of data. It can be used for any type of database where knowledge of some data (of sufficient size) is required for a lookup to be possible. In our setting, this “key information” is the results of some fixed forensic tests.

Two recent papers that are related in a technical sense are those of Monrose, Reiter and Wetzel [23] who used similar methods for strengthening login sessions by measuring typing speeds; and Ellison, Hall, Milbert and Schneier [10], who suggested a method for “fuzzy” passphrase based logins, where only a certain portion of the passphrase needs to be correct. Whereas the former employs techniques believed to be resistant against lattice-based attacks (see e.g., [7, 25, 33]), the latter has been found to be susceptible to such attacks, as recently pointed out by Bleichenbacher [4]. Note that in both cases, the technology is in some sense used to relate attributes of an individual (typing habits and personal information, respectively) to the identity of that individual.

Our solution is in principle related to that of Monrose, Reiter and Wetzel, and one of our proposed plug-ins uses a similar structure to embed secret information. It differs on a few important points, and to understand these, we will briefly review how the MRW scheme for password hardening works. The scheme is based on a password, a secret sharing scheme such as polynomial interpolation, and a two-dimensional matrix with two columns and some number of rows, where each entry may contain a point useful for interpolation or a random number. Each row corresponds to some measurement of how the user typed a letter of the password. For example, one column might correspond to the letter being typed “quickly” after the previous letter, while the other column corresponds to it being typed “slowly” (as compared to some average speed). The idea is that individuals routinely type *certain* letters in their password either quickly or slowly. When an individual routinely types a letter slowly, a share of the secret is entered into the first column, and similarly for the second column if the user routinely types that letter quickly. In either case, the other entry in the row contains a random number. If an individual does not have a routine behavior for a measurement corresponding to a particular row in the table, shares of the secret can be entered in both columns of that row. Once the table is filled, each entry in the table is further encrypted with the user’s password.

This table is used to harden the user’s password as follows. While the user enters his password, the system measures the time between and duration of keystrokes. If a measurement falls below a certain threshold for the i th keystroke, value $(i, 1)$ from the matrix is selected; otherwise, value $(i, 2)$ is selected. Each such selected value (the number of which equals twice the length of the password, minus one) is decrypted, using the entered password as the decryption key. Then the values collected are used to determine the secret of the secret sharing scheme. For example, when using polynomial interpolation for secret sharing, the values are interpreted as points which can be thought of as lying on a polynomial. These points are interpolated to determine where the polynomial intersects the y-axis, and the resulting value is treated as a key. Using this key, a file (containing

keystroke statistics) is decrypted. If the result is a file of the correct format (which can be obtained by introducing some redundancy), the login session is said to succeed. The measurements taken of the user's typing speeds are then added to the statistics in the file. A new table and polynomial are created based on the modified statistics, and the statistics file is encrypted with the key from the new polynomial and again stored in the user profile.

The MRW technique shares with ours the use of personal information (typing patterns vs. genetic profiles) to allow access to some resource or data, the mapping of features of personal information to points on a polynomial, and the use of polynomial interpolation as a secret sharing scheme in the decryption process. However, the situations differ in two principal ways: 1) All of the personal information used in the MRW scheme is available at each use since the user must type every key of the password. In one of our proposed solutions, substantial portions may be missing, and the scheme works as long as a sufficient amount is known. 2) The bulk of the entropy in the MRW scheme resides in the password, and the personal information used is related to a particular password. Thus, the key can easily be reset by choosing a new password. In our case, the key is genetic information which cannot be reset, and thus must remain secure for the lifespan of individuals in the database.

A third recent paper of relevance is that of Juels and Wattenberg [15], in which they put forth a method for hash functions whose inputs can contain a certain number of errors without changing the output. However, if the errors exceed a certain threshold, the resulting output is unpredictable, as for standard hash functions. We show how to employ this method as a building block in one of our proposed schemes, and discuss it further in that context.

4 Model and Problem Definition

In this section we present a trust model for a forensic DNA database, and define the problem of implementing such a database in a secure manner. Our definition hinges on two parameters: (1) the number of tests required for a lookup and (2) the number of such tests that may be in error. We also give an overview of our correctness criteria, and discuss the issue of how many DNA tests would actually be required for such a scheme to be secure.

4.1 Trust Model

In order to perform the tests required to enter an individual into a forensic DNA database, it is necessary to obtain a blood or tissue sample from the individual, and to submit that sample to a government or commercial lab for genetic testing. Later, when a lookup is to be performed, similar testing has to be performed on the blood or tissue sample for which a lookup is to be made. In our model, we have to trust that the labs performing these tests are honest, and that almost all of the test results are correct. Additionally, we trust the labs not to run inappropriate tests on the samples, not to reveal information that would compromise

the privacy of the individuals from whom the samples were taken, and to discard samples and test results after their test results have been reported. It is possible to conceal the identity of an individual from a lab (that does not violate the above assumptions) simply by using an alias identifier instead of a personal identifier to label the samples. This is already commonly done in genetic testing so that control samples may be used to verify the accuracy and integrity of the lab processes [24].

In addition, we have a database producer whose task it is to administer the collection of blood or tissue samples, send these to the above mentioned labs, collect the results, and create database entries. This database producer may consist of a multitude of independent entities, each responsible, for example, for a certain geographical region. We trust each such entity not to enter incorrect data, and not to violate the privacy of the individuals in the database.

4.2 Problem Definition

In the following, T represents the trusted authority described above, and h is a hash function that can be described as a random oracle [3]. The goal of an attacker in the following discussion is to obtain some plaintext information from some record in the database. (We do not require that the attacker succeed in decrypting an entire entry, but merely that some non-trivial information is obtained about some plaintext record.)

A problem instance is characterized by three parameters, k , c and s , and a set of n different tests τ_1, \dots, τ_n . The parameter k refers to the number of test results (out of n) which must be provided as part of a query, c refers to how many of these test results may be in error, and s is a security parameter such that 2^{-s} is the maximum success probability of an adversary. The test results τ_j are typically small integers; for simplicity we denote the range of possible outcomes of test τ_j as $[1, \dots, \omega_j]$. Let M_i be a plaintext message describing the individual with index i . For example, M_i might be a social security number, a reference to an entry in some database of convicted offenders, or a lab name and lab-assigned individual identifier as in the CODIS system. A record in the database is comprised of an encrypted version of M_i , possibly along with certain additional information specific to a particular problem solution.

A query is a set of test results from an unknown individual u . While the test results may be for any k -subset of the n tests in the system, we refer to them as $\tau_{u1}, \dots, \tau_{uk}$ for denotational simplicity. A query result is a plaintext M_u if a match is made, and a string stating "no match" otherwise. A scheme is said to be *secure* if it meets the following requirements:

- **Correct.** When at least k test results are presented as part of a query, if all the computation is performed as stated in the protocol description, the query result is M_i for an individual i whose test results match the query values on all but at most c tests, but for a negligible probability.
- **Complete.** All the required computation can be performed in time polynomial in the length of the security parameters.

- **Hiding.** It is impossible to extract data from the database without prior knowledge of test results of the individual for whom the lookup is to be performed.

4.3 Parameter Choices

In this section, we discuss the selection of k . We do not assume that the tests are identically distributed, although we do assume that they are independent (the case where tests are correlated can be handled with obvious extensions). In all cases, the minimum entropy present in some $k - c$ tests supplied as part of a query, and not used for error correction or indexing, equals the security parameter s .

We have computed the test-result entropy present in frequency distributions given in [11] for two loci, d21s11 and HumFGA. The entropy was computed using the formula $E = -\sum_{i=1}^{\omega_j} f(i) \cdot \log_2 f(i)$ where $f(i)$ is the frequency with which the test takes on allele value i . Since some alleles are more common in certain sub-populations, frequency distributions for tests are computed independently for the different ethnic groups in [11]. Assuming that an attacker may know the ethnic group of an individual, we conservatively take the *minimum* of the computed sup-population entropy values as the entropy of the test. For these two tests, and assuming independence of the two alleles, the minimum entropy for any of the given ethnic groups is 5.3 and 5.24 bits respectively. If we assume this is representative for sites used in modern test protocols, and that a security parameter of 80 bits of entropy is desirable, then approximately 80/5.24, or a minimum of 15 tests, would be required.

5 Solutions

We begin by presenting a general solution framework, and then show how the scheme can be specialized for three different cases. The solution corresponding to the first case is the simplest and easiest to analyze. However, it assumes that individual tests produce accurate results and that the set of tests is fixed and unchanging. The second solution is almost as efficient, and allows for some errors to be corrected using standard error correcting methods. Finally, the third solution, which bears resemblance to the MRW solution, allows for the query set of tests to differ from the set of tests originally run on the individual, as long as the intersection of these sets contains k or more of the n tests. This solution also allows for limited error correction when more than k tests are available.

5.1 Solution Framework

Our database is produced by the trusted authority T , and then sent to individual control offices, such as local or regional police headquarters. In the following, E_{K_i} is a symmetric encryption function with key, K_i , chosen uniformly and at random

from a large keyspace \mathcal{K} . The cipher is assumed to have the property that given (only) one ciphertext and a potential plaintext, it is not possible to determine if these correspond to each other without knowledge of the key. This property relates directly to the unicity distance of the cipher (see [22] for an overview).

Each record in the database consists of a form of M_i encrypted with E_{K_i} (discussed further below), and a “lock” L_i . K_i may be randomly generated or may be computed from the tests, τ_{ij} during initial record creation. Later, when attempting to determine if a query set matches this record, a “candidate” \tilde{K}_i is computed, making use of the test results in the query set and the lock, L_i .

Creating a New Record.

1. **Identification.** For each new individual, indexed by number i , the trusted authority T obtains a blood or tissue sample, and performs the n tests on the sample, determining the outcome τ_{ij} , $1 \leq j \leq n$, of each such test.
2. **Secret key and lock generation.** T generates a “lock” L_i , and a key $K_i \in \mathcal{K}$. Here, K_i can (only) be computed given the lock L_i and the test values τ_{ij} , which act to “unlock” L_i . The manner in which these parameters are generated is scheme-specific, and will be detailed in Sect. 5.3.
3. **Data encryption.** Let M_i be a plaintext message describing the individual with index i . Let $\mu_i = M_i | red(M_i)$ for some fixed redundancy function red . T computes the ciphertext $c_i = E_{K_i}(\mu_i)$ of μ_i . T further computes L_i in a scheme-specific manner.
4. **Record creation.** T appends the record $R_i = (c_i, L_i)$ to the public database, erases all intermediary values, and discards the remainder of the blood or tissue sample.

We note that as new database entries are created, these records can simply be appended to the existing database.

Searching the Database.

1. **Identification.** Say that an agent, A , wants to look up a given individual in the database. A obtains a blood or tissue sample, presumably left at the scene of a crime, from an unknown person u . A has k of the n determined tests performed on the sample, say tests 1 through k for denotational simplicity, giving results $\tau_{u1}, \dots, \tau_{uk}$. These results comprise the query set for the database.
2. **Data decryption and output.** For each record $R_i = (c_i, L_i)$ of the database, the agent computes \tilde{K}_i in a scheme-specific manner using L_i and $\tau_{u1}, \dots, \tau_{uk}$, and attempts to decrypt the corresponding ciphertext, c_i , using \tilde{K}_i . The result is $\tilde{\mu}_i = D_{\tilde{K}_i}(c_i)$. If this has the correct form (i.e., is of the form $M_i | red(M_i)$) then A outputs (i, \tilde{K}_i, M_i) . With overwhelming probability, $i = u$ when a match is made. If no record results in a transcript of the correct form, then A outputs “no match” and halts.

5.2 Indexing

As discussed so far, the scheme requires that each record in the database be scanned during a lookup. While a match against a specific record will be very fast compared, say, to a fingerprint match, it remains worthwhile to consider how to index the database for quicker lookups. One possibility of indexing would be to separate the test results into two portions. The first portion would be set aside for encryption and matching, as previously explained, whereas the second portion would be used to generate an index of the records to be created. Given test results, we would select the database entries whose indices are closest to that generated from the results. The second portion could simply be left as plaintext, as it does not reveal any information about the owner, as long as the identity of the record owner remains unknown.

We note that this technique allows a tradeoff between the cost for performing experiments and the computational expense, as we can narrow down the search by determining partial information about the index. However, by keeping the result of additional tests in plaintext, it becomes important that it not be possible to match records in the database with individuals. For example, an attacker might attempt to correlate new records appearing in the database with known convictions in a given area. This possibility can be mitigated by batching updates, or by padding small updates with false records, etc.

5.3 Three Plug-ins

We consider three ways to construct the building block referred to as “the lock” in the above described scheme.

Fixed-query Solution. In our first implementation, we assume that the tests τ_1, \dots, τ_n are fixed and *universal*, meaning that the same set of tests will always be available whenever entries are created or samples tested from crime-scenes. We assume further that almost no errors are introduced when performing the tests. In this solution, we do not need any lock L_i , but plainly set the key K_i (resp. \tilde{K}_i) as an appropriately long portion of $h(\tau_{i1}, \dots, \tau_{in})$ for the n tests performed. Assuming that the hash function can be modelled as a random oracle, it is sufficient that our tests have a sufficient entropy (say 80 bits) for the resulting scheme to be secure. We note that this solution generalizes directly to a small number of distinct *sets* of standard tests, corresponding, for example, to standard sets of loci used in different countries. This can be achieved plainly by computing several such keys, and creating several records correspondingly.

Fixed-Query Solution with Error Correction. This technique incorporates the use of *fuzzy commitments* from a recent result by Juels and Wattenberg [15] to allow some portion of a key to be in error and yet be successfully used to decrypt a codeword. Recall that error correction codes correct messages³ to the

³ In this context, “message” refers to the key which may be in error, not to the plaintext object being encrypted/decrypted.

nearest *codeword*. The set of messages which will be corrected to a codeword is by definition the *neighborhood* of that codeword. Without reference to a particular error-correcting code, we outline how the technique from [15] may be used to encrypt some object μ_i , based on some “user key,” U , such that any \tilde{U} can be used to decrypt μ_i as long the difference between \tilde{U} and U is small – within the power of the error correcting scheme to correct. In general, U can be thought of as biometric information such as a fingerprint – in our setting, U would be the $\tau_{i1}, \dots, \tau_{in}$.

The first step of encrypting μ_i based on U in this scheme is to select a codeword, σ_i , at random from a set of codewords of the same length in bits as U in the chosen error-correcting code. The next step is to compute the *bitwise difference*, δ_i , between U and σ_i . An appropriate portion of $h(\sigma_i)$ is then used as K_i , and μ_i is encrypted using E_{K_i} to form c_i . The value, δ_i is stored as plaintext with c_i .

In order to later decrypt c_i with a candidate key, \tilde{U} , δ_i is subtracted in a bitwise manner from \tilde{U} . Error correction is applied to the resulting $\tilde{U} - \delta_i$ value to generate $\tilde{\sigma}_i$, the closest codeword under the error correction code. If the difference between U and \tilde{U} is small enough, this correction will succeed giving $\tilde{\sigma}_i = \sigma_i$. The candidate key, \tilde{K}_i , is set as an appropriate portion of $h(\tilde{\sigma}_i)$, and an attempt is made to decrypt c_i with \tilde{K}_i . We can use this construction in our scheme, setting $L_i = \delta_i$.

Flexible-Query Solution. This technique adapts secret sharing techniques [35] to allow a record to be decrypted with any number of tests missing, as long as sufficient tests are presented for the security of the scheme. The idea is that the lock, L_i allows test results to be mapped to points on a polynomial of degree k , allowing interpolation of K_i if at least k points are presented.

In this solution, p and q are large primes such that $q|p-1$, and g is a generator of a subgroup G_p of size q . Consider first the generation of the key K_i and the lock L_i on record creation:

T generates a random k -degree polynomial $\rho_i(x)$ with coefficients in Z_q . T further computes a set of n points, $s_i(j) = \rho_i(h(j, \tau_{ij})) \bmod q$, $1 \leq j \leq n$, and a second set of n points, $P_i(j) = g^{s_i(j)} \bmod p$. The lock, L_i is set to $P_i(1), \dots, P_i(n)$. K_i is computed as an appropriately long portion of $h(g^{\rho_i(0)} \bmod p)$.

In order to perform a look-up, $\tilde{P}_i(0)$ is extrapolated using the measurements τ_{uj} and the points $P_i(j)$ on the polynomial as follows: Let $x_{uj} = h(j, \tau_{uj})$ be the x coordinate of such a point, and \mathcal{X} the set of such coordinates, where this set is of size k . The interpolation is computed as $\tilde{P}_i(0) = \prod_{j=1}^k P_i(j)^{\lambda_{ij}} \bmod p$, where λ_{ij} is the Lagrange weight for the coordinate x_{ij} in \mathcal{X} . Finally, \tilde{K}_i is computed as the appropriate portion of $h(\tilde{P}_i(0))$.

Discussion. Figure 1 illustrates a portion of the construction of K_i and L_i . The idea is to use the secret of a secret sharing scheme [35] as the key K_i . By mapping test results to shares of the secret, we allow decryption of the record

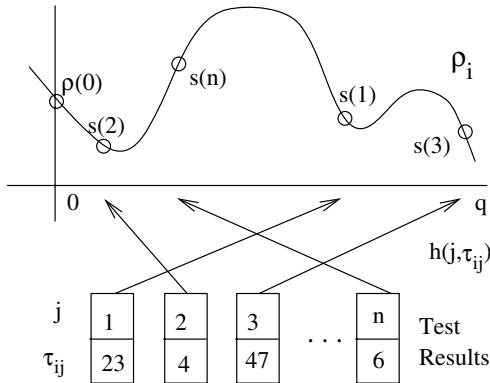


Fig. 1. Relationship between ρ_i , τ_{ij} and $s_i(j)$

if and only if sufficient tests are present. In this case, the secret sharing scheme is based on interpolation of a discrete polynomial, ρ_i . As shown in the figure, h is used to map (test name, test result) pairs into points on the polynomial. One possibility would be to use these $s_i(j)$ values as the lock L_i . If this were done however, lattice based attacks [33] might be used to determine ρ_i based on the limited set of points which might be generated from the test results. By using h to perturb the location of the points on the x -axis, such attacks are made more difficult. However, the final step of our lock construction, which uses $P_i(j) = g^{s_i(j)}$ in the lock rather than $s_i(j)$ itself, provides much greater protection by destroying the homomorphic properties of the system – for example, it is not possible to compute a representation of the product of two s_i values when they are represented as exponents of g , if the Diffie-Hellman assumption holds.

It is clear that this scheme allows flexibility in which tests are used, since k can be less than n , so long as any k tests contain at least s bits of entropy (recall that s is the security parameter of the problem instance). This is in contrast to the previous two solutions which require $k = n$.

Error Correction. In order to allow for error correction in this scheme, we can try different subsets of k test results for a match. We note that this solution is worst-case exponential in the number of errors present, but can be used for some low number of incorrect tests, particularly if indexing is employed (as this narrows down the search space substantially), and is much faster than trying every possible *value* of an incorrect test.

Remark Concerning Entropy. We note that in the last solution, we want the different tests results to have similar entropy, as an attacker only needs to successfully guess k of these, and the success rate of such an attack relates only to the entropy of the tests being guessed. If the tests have substantially different entropy, we can split test results with large amounts of entropy into several log-

ical tests, causing such logical tests to have similar entropy. (Clearly it may not be possible to utilize all the entropy present in all tests if doing so.)

6 Security Analysis

It is clear that our scheme is *correct* (i.e., that the correct output is obtained) and *complete* (i.e., that all computation can be performed.) Moreover, our scheme is *hiding*, i.e., it is impossible to extract data from the database without first performing a sufficient number of tests on tissue or blood. More formally,

Claim: Assume that K_i is set as an s -bit number chosen uniformly at random. Let \mathcal{A} be an adversary attempting to decrypt a ciphertext c encrypted with the encryption method E , using this unknown key K_i , which is only used for the creation of c . Then, our scheme is hiding against \mathcal{A} , or \mathcal{A} can be used as a black box algorithm to break the assumed semantic security of E .

This follows straightforwardly from the definitions of the related concepts. The security of our scheme follows from the above, given that the hash function h can be modeled as a random oracle, and the record lock L_i does not leak any information that can be used to compute the key K_i . (The lock is the only auxiliary data related to the key K_i , under the assumption that there is no correlation between the test results used for the lock creation and those used for indexing.)

It is clear that our solution, if employing the first plug-in, is secure, given that we use no lock in this plug-in. In the second solution, it is clear that bits proportional to the the distance between codewords in the error correcting code may be leaked, since the lock, while not revealing the codeword to which the user key is mapped, may reveal the user key's relative position to the nearest codeword. The precise amount of leakage, and thus a precise analysis of the security of the scheme, is dependent upon the particular error correction code chosen, and the selection of an appropriate code for genetic tests is a topic of ongoing research. Further discussion of the scheme's security may be found in [15]. Whereas we know of no proof of security for our third construction, we note that this solution avoids the properties employed by lattice-based attacks [33], as it avoids linear relationships between the items used for interpolation.

6.1 Attacks with Biological Knowledge

One might argue that a potential attacker with access to other databases of genetic information would be able to perform a successful search in the database we are designing. After all, if a database with complete genetic information about all human beings (of which there are only about 2^{33} , an insignificant number in comparison to the security parameters of the scheme) is available to the attacker, this data can be used to limit search of the key space to only existing combinations, allowing decryption of all of the information present in our database. Such an attack is analogous to dictionary attacks on user passwords.

In general, we consider the possession of such a database by an attacker to be in itself a tremendous risk to the genetic privacy of the individuals whose information appears therein. It is unlikely that the additional ability to decrypt a forensic database would add significantly to that risk. However, one might argue that an *anonymized* genetic database would not constitute such a risk to individual privacy. If such an anonymized database included the test loci used in a forensic database organized as we propose, the forensic database would serve exactly to reveal the identities associated with records in the database. In order to mitigate this risk, loci used for forensic purposes should simply be left out of any such large-scale anonymized database.

Remark - the Icelandic Database. In fact, just such a database is being constructed in Iceland for the purpose of medical genetic research [27], and is the subject of significant controversy on exactly the issue of genetic privacy (see, for example, [19]). We argue that this database presents a far greater risk to individual genetic privacy than is posed by publication of any present or conceived forensic DNA database, simply because, as already mentioned, the data stored in forensic databases is not known to relate to any individual attribute. By contrast, the data to be gathered in the Icelandic database is exactly related to genetic disease, so that the database will be useful for medical research. Regarding the possibility of using a forensic database to destroy the anonymous property of a database such as the Icelandic one, it was pointed out in [1] that many other avenues exist for determining the identity of individuals in such a database, using features such as the number and gender of relatives.

7 Conclusions and Future Work

In this paper we introduce a cryptographic approach to protecting the privacy of information in forensic DNA databases. Our solution allows the recovery of an individual's identity and/or criminal record if – but only if – the genotypes present at a certain number of loci on the suspect's DNA are determined. Based on differing assumptions about standardization of forensic genetic tests and the need for error correction, we have proposed three solutions to this problem within a common framework. In our first, fixed-query solution, tests are completely standardized and no error correction is required. Our second solution extends the first to add error correction using fuzzy commitments from [15]. Our third solution employs polynomial interpolation to allow decryption based on a threshold number of tests, allowing arbitrary tests to be missing. In each case, a “lock” is stored with each record which, when combined with test results for the individual, allows the individual's identity to be decrypted. In each case, if the requisite loci are not identified, it is not feasible to identify or decrypt the record in question. Given the assumption that deriving the threshold number of values can only be accomplished by testing a sample of biological material from the individual, it is not feasible to derive any genetic information from the database which is not already available in the sample.

Based on preliminary computations about the entropy present in modern tests, current profiles may not contain sufficient entropy for achieving sufficient security, implementing error correction, or adding indexing. However, with the rapid introduction of new sites and loci, it is reasonable that a profile with over 100 bits of entropy would be easily available, or may in fact already be available at the time of publishing.

While we argue that these techniques are practical from a computational perspective, and will soon be practical from a genetic testing perspective, certain limitations remain that are the topic of ongoing research. First, it may not be practical to perform all possible tests against each individual. In itself, this is not a problem, unless the amount of biological material available in a sample is so small as to strictly limit the number of tests which may be performed on the sample. In this case, it would be useful to know which test to run next given the output of previous tests. For example, given the result of a “universal” test given to all individuals, one would like to run subsequent tests which would help distinguish between those records in the database which matched the sample on this test. Unfortunately, it is not obvious how to do this without revealing secret information. Second, there are certain searches which may be performed on an unencrypted database which may not be performed on our database. One such search is for a close relative of an individual from which a sample is obtained. In the case of a parent or child, one would expect about half the tests to match, which may provide useful leads in a case. Another such search takes place when a blood sample contains a mixture of blood from two individuals, where the identity of neither individual is known to the investigators. In this case, exact values of test results are not known, since each test returns results for both individuals.

We note that it may be possible to reuse the information of the offset described in the second plug-in for purposes of indexing. We have not examined how to practically do this.

Finally, we note that the general form of the problem definition and our solutions apply to any problem domain where a *secure* but *imprecise* key is required. These problems include the use of biometric data (fingerprints, retinal scans, etc.) [15] and the generation of keys which can be effectively recreated by human beings [10].

Acknowledgements. We thank Daniel Bleichenbacher and Phong Nguyen for helpful discussions on lattice based attacks. We thank Ari Juels for helpful general discussions, and Amin Shokrollahi for explanations of error correcting codes.

References

1. R. Anderson.: The DeCODE proposal for an Icelandic health database. <http://www.cl.cam.ac.uk/~rja14/iceland/iceland.html>, 1998. 387
2. G.J. Annas.: Privacy rules for DNA databanks: Protecting coded ‘future diaries’. *JAMA*, 270(19):2346–2350, 1993. 374

3. M. Bellare, P. Rogaway.: Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 62–73. ACM Press, 1993. **375, 380**
4. D. Bleichenbacher.: Private communication. **378**
5. B. Budowle, T.R. Moretti.: Genotype profiles for six population groups at the 13 CODIS short tandem repeat core loci and other PCR-based loci. *Forensic Science Communication*, 1(2), July 1999. **375**
6. D. Butler.: UK to set up DNA database of criminals. *Nature*, 370:588–589, 1994. **377**
7. M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.-P. Schnorr, J. Stern.: Improved low-density subset sum algorithms. *Journal of Computational Complexity*, 2:111–128, 1992. **378**
8. A. de Gorgey.: The advent of DNA databanks: Implications for information privacy. *American Journal of Law and Medicine*, 16:381–398, 1990. **374**
9. C. Dib, S. Faure, C. Fizames, D. Samson, N. Drouot, A. Vignal, P. Millasseau, S. Marc, J. Hazan, E. Seboun, M. Lathrop, G. Gyapay, J. Morissette, J. Weissenbach.: A comprehensive genetic map of the human genome based on 5,264 microsatellites. *Nature*, 380:152–154, 1996. **375**
10. C. Ellison, C. Hall, R. Milbert, B. Schneier.: Protecting secret keys with personal entropy. *Future Generation Computer Systems*, to appear, 1999. **378, 388**
11. R. Fourney.: Allele frequency distribution tables. <http://www.cstl.nist.gov/div831/strbase/freq.tab.htm>. **381**
12. K. Fox.: Criminal justice. In *Mapping Public Policy for Genetic Technologies*. National Conference of State Legislators, 1998. **373**
13. C. Goldberg.: DNA databanks giving police a powerful weapon, and critics. *New York Times*, Thursday 19, 1998. **373, 374**
14. B.E. Henry, G.S. Rogers, C. Mauterer, D.K. Dodd, J.W. Hicks.: Technical evaluations of databanking methods. In *Proceedings of the Eighth International Symposium on Human Identification*, 1997. **373**
15. A. Juels, M. Wattenberg.: A fuzzy commitment scheme. In *6th ACM Conference on Computer and Communication Security*, (to appear), 1999. **375, 379, 383, 384, 386, 387, 388**
16. L.T. Kirby.: *DNA Fingerprinting: An Introduction*. Oxford University Press, 1992. **376**
17. R. Köttger.: Probe nummer 3889 führte zum Mörder. *Die Welt*, August 27, 1999. **374**
18. T.G. Krontiris.: Minisatellites and human disease. *Science*, 269:1682–1683, 1985. **377**
19. Mannvernd.: The Mannvernd web site. <http://www.mannvernd.is>. **387**
20. J.E. McEwen.: DNA data banks. In M. Rothstein, editor, *Genetic Secrets: Protecting Privacy and Confidentiality in the Genetic Era*, pages 231–254, 1997. **374, 377**
21. J.E. McEwen, P.R. Reilly.: A review of state legislation on DNA forensic data banking. *American Journal of Human Genetics*, 54:941–958, 1994. **373, 377**
22. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone.: *Handbook of Applied Cryptography*. CRC Press, 1997. **382**
23. F. Monrose, M. Reiter, S. Wetzel.: Password hardening based on keystroke dynamics. In *6th ACM Conference on Computer and Commnication Security*, (to appear), 1999. **378**
24. National Research Council.: *The Evaluation of Forensic DNA Evidence*. National Academy Press, 1996. **376, 377, 380**

25. P. Nguyen, J. Stern.: The hardness of the hidden subset sum problem and its cryptographic implications. *Crypto'99, LNCS 1666*, pages 31–46, 1999. [378](#)
26. S.J. Niegzoda Jr., B. Brown.: The FBI laboratory's COmbed DNA Index System program. In *Proceedings of the Sixth International Symposium on Human identification*, 1995. [373](#), [377](#)
27. Working Group of the Ministry of Health and Social Security.: Bill on a health sector database. <http://brunnur.stjr.is/interpro/htr/htr.nsf/pages/gagnagr-ensk>, 1998. [387](#)
28. E. Peerenboom.: Central criminal DNA database created in Germany. *Nat Biotechnol*, 16(6):510–511, 1998. [377](#)
29. R. Perez-Pena, J. Blair.: Albany plan widely expands sampling of criminals' DNA. *New York Times*, Saturday, August 7, 1999. [374](#)
30. P.R. Reilly.: DNA banking. *American Journal of Human Genetics*, 51:1169–1170, 1992. [374](#)
31. P.R. Reilly.: Fear of genetic discrimination drives legislative interest. *Human Genome News*, 8:3–4, 1997. [374](#)
32. B. Scheck.: DNA data banking: A cautionary tale. *American Journal of Human Genetics*, 54:931–933, 1994. [374](#)
33. C.-P. Schnorr, H.H. Hörner.: Attacking the Chor-Rivest cryptosystem by improved lattice reduction. *Eurocrypt'95, LNCS 921*, pages 1–12, 1995. [378](#), [385](#), [386](#)
34. J.W. Schumm.: New approaches to DNA fingerprint analysis. *Promega Notes Magazine*, 58:12–18, 1996. [375](#)
35. A. Shamir.: How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. [375](#), [384](#)
36. G.R. Sutherland, R.I. Richards.: Single tandem DNA repeats and human genetic disease. In *Proceedings of the National Academy of Science USA* 92, pages 3636–3641, 1995. [377](#)
37. Technical Working Group on DNA Analysis Methods (TWGDAM).: The combined DNA index system (CODIS): A theoretical model. In L.T. Kirby, editor, *DNA Fingerprinting: An Introduction*. Oxford University Press, 1992. [374](#)
38. K. Wrogeaman, V. Biancalana, D. Devys, G. Imbert, Y. Trottier, J-L. Mandel.: Microsatellites and disease: A new paradigm. In *DNA Fingerprinting: State of the Science*, pages 141–152, Basel, Switzerland, 1993. Birkhäuser Verlag. [377](#)

Making Hash Functions from Block Ciphers Secure and Efficient by Using Convolutional Codes

Toru Inoue¹ and Kouichi Sakurai²

¹ Advanced Mobile Telecommunications Security
Technology Research Laboratories Co., Ltd
BENEX S3, 3-20-8, Shin-Yokohama, 222-0033 JAPAN
t-inoue@amsl.co.jp

² Kyushu University, Dept. of Computer Science
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. We improve Knudsen-Preneel's constructions for cryptographic hash functions based on block ciphers with error correcting codes. We first modify to extend original constructions, which are effective only for non-binary codes, to the case with binary codes (e.g. BCH codes). We also revise the original method by introducing convolutional codes, whereas the previous adapts only block codes. This reduces the circuit complexity of the hardware-implementation $1/N$ times in terms of the number of (Davies-Meyer's) module functions than that based block error correcting codes.

Key words: *hash functions, block ciphers, error correcting codes, BCH codes, convolutional codes, Collision resistant compression*

1 Introduction

Hash function. Hash functions are important cryptographical techniques for making signature, or building public key crypto-systems. Currently used hash functions include SHA, SHA-1, MD-4, MD-5, etc. Although methods which use particular algorithms for these hash functions are reasonably speedy, some of the methods are known to vulnerable to certain types of attacks[Dob96a,Dob96b]. Meanwhile, a Davies-Meyer method is known as a method of obtaining a hash function through a repeated use of a block cipher as a compression function. An advantage of using a block cipher to a hash function is that if the security of the block cipher is guaranteed, the hash function using the block cipher is secure[KP96,KP97]. In addition, the security will not be lost even by an expansion of a single hash mode into a multiple hash mode[KP97]. In this paper, we assume to use a block cipher whose security is guaranteed. Moreover an error correction coding for data protection is easily adopted using same error correction encoders while encrypting data. We also assume the following hypotheses:

1. Hash round function is secure.
2. A short cut which breaks the single hash mode does not exist.
3. If the security of the block cipher is guaranteed, the security of the hash function using the block cipher is also guaranteed.
4. A document is sufficiently long.

Despite this, if we use an ordinary 64-bit block cipher as it is, a collision (or a birthday attack) occurs for every 2^{32} trials on the average, which is not highly secure anymore in the present situation regarding ciphers.

Knudsen-Preneel's approach and the challenging problems. The construction proposed by Knudsen and Preneel is known as a method which uses error correcting codes with the multiple Davies-Meyer functions for enhancing a collision resistance[[KP96](#),[KP97](#)]. Their method requires to use quaternary (n, k, d) linear codes, where the symbol n denotes a code length, the symbol k denotes an information symbol number, and the symbol d denotes a minimum distance, and improves a collision resistance from $2^{m/2}$ to $2^{(d-1)m/2}$. However, Knudsen and Preneel's method requires to use only non-binary codes in the construction. For design flexibility we want to use binary codes, too. Another problem is when we use block codes, as a tendency, the code length becomes longer to increase the number of error correcting capability, the computation accordingly becomes more complex. In addition, many Davies-Meyer modules are required because of a long code length, then it is necessary to prepare a number of apparatuses for realizing Davies-Meyer functions. Thus, there is a room for improvement of Knudsen-Preneel's approach with respect to the efficiency for hardware constructions.

Our Contribution

Using binary codes. We first try a similar construction as Knudsen-Preneel by using binary codes, such as BCH codes. The direct method by Knudsen-Preneel fails in the case of binary codes. Then, we revise Knudsen-Preneel's construction for adapting the use of binary codes. However, this revision with binary codes are not so efficient as the Knudsen-Preneel's original with non-binary MDS codes: our hash rate becomes very low. Next, we devise efficiency by error correction encoding only message vectors, while our previous method encodes not only message vectors but also key vectors. Though it requires stronger (but still reasonable) assumption than that of Knudsen-Preneel, our second construction achieve good hash rate.

Using convolutional codes. We try to resolve the second problem by using convolutional codes. Our proposed method requires to select the number of multiple Davies-Meyer functions, to the same as the sub-block length n_0 of a convolutional code and to enter inputs in N time units where N is a constraint length, to thereby reduce the size of a Davies-Meyer function down to the sub-block length n_0 of a convolutional code from a code length n which is a code length in a case

where a block code is used. This method using convolutional codes reduces hardware $1/N$ in terms of the number of Davies-Meyer functions than where block error correcting codes are used under the same functional conditions. For example, the construction using $(15,7,5)$ BCH codes require 15 Davies-Meyer functions. However, the construction using convolutional codes such as $(3, 2, 5; N = 14)$ CSOC, requires only three Davies-Meyer functions, namely, convolutional codes reduce the number of Davies-Meyer functions by the factor of $1/5$.

2 Hash Function

In a telecommunication system in which messages, data and the like are encoded and transmitted as cryptograms to protect the confidentiality, a hash function, i.e., a compression function for compressing and signing a message, is used. To compress a document of an optional length into a certain predetermined length, cryptographic hash function is used. For example, in order to sign using the DSS (Digital Signature Standard), a document of an optional length is converted using a hash function into a hashed value of a 160-bit block once, and a signature of 320 bits, for instance, is added to the 160-bit block hashed value. The hash function needs be devised so as to obviate a collision. A collision is an event that $h(x) = h(x')$ holds when $x \neq x'$. According to the definition of a resistance of a hash function against a collision, there are a weak resistance and a strong resistance. A weak collision resistance is called a preimage resistance or a 2nd-preimage resistance.

A preimage resistance expresses to what extent it is difficult to find x' which converts into $h(x')$ in relation to an inputted hashed value $H = h(x)$. A 2nd-preimage resistance expresses to what extent it is difficult to find a second input x' which satisfies hashed value $H = h(x) = h(x')$ in relation to the input x . That is, when we have a document x and a corresponding hashed value $h(x)$, if it is difficult to find a document x' which is converted into the same hashed value $h(x)$ whatever the document x is, and further, if it requires W trials on the average to find such a document x' , a 2nd-preimage resistance of the hash function h is W . In this paper, a preimage and a 2nd-preimage will not be distinguished from each other, but instead, treated equally.

A strong collision resistance is called a collision resistance or a resistance against a birthday attack. In short, a strong collision resistance expresses to what extent it is difficult to find any input pair $(x, x'; x \neq x')$ which converts into $h(x) = h(x')$. More precisely, if W trials on the average are necessary to find a pair of different documents having the same hashed value, a resistance against a birthday attack of the hash function is W . A collision resistance normally means a strong resistance. A hash rate of a hash function based on an m -bit block cipher is defined by the number of m -bit message blocks which are processed during one encryption or decryption. The method using block cipher is called a Davies-Meyer method [MMO85,DP84]. An encryption algorithm for an m -bit block cipher is denoted at the symbol $E_K(x)$, and its m -bit key is denoted at the symbol K . The compression function is called a Davies-Meyer function.

We consider iterated hash function based on an easily computable compression function $h(\cdot, \cdot)$ originated from two binary sequences of lengths m and l to a binary sequence of length m . The message M is split into blocks M_i of l bits, $M = (M_1, M_2, \dots, M_n)$. If the length of M is not a multiple of l , M is padded by using an deterministic padding rule. The hash value H_n of length m is obtained by computing iteratively,

$$H_i = h(H_{i-1}, M_i) \quad i = 1, 2, \dots, t \quad (1)$$

where H_0 is an initial value, denoted by IV , namely $H_0 = IV$. The function $h(\cdot, \cdot)$ is called hash round function. Hash result

$$\text{Hash}(IV, M) = H_t \quad (2)$$

is obtained by repeating calculation (1). To relate the security of $\text{Hash}(\cdot)$ to that of $h(\cdot, \cdot)$, we needs to append an additional block at the end of the input string concerning its length, as MD-strengthening leading to the following result [Dam89,Me89].

Theorem-MD: Let $\text{Hash}(\cdot)$ be an iterated hash function appended MD-strengthening. Then preimage and collision attacks on $\text{Hash}(\cdot, \cdot)$ have roughly the same complexity as the corresponding attack on $h(\cdot, \cdot)$ [KP97]. In practical applications, the IV of a hash function is fixed in the specifications. This leads to a higher security level, so Theorem-MD gives a lower bound on the security of $\text{Hash}(IV, \cdot)$ [KP97].

ASSUMPTION 1 [KP96,KP97]: Encrypting (of the m -bit block) about $2^{m/2}$ times is necessary to find a collision to h as far as a secure block cipher is used, and encrypting about 2^m times is necessary to find a preimage to h .

The message M_i , the hashed value H_i and the immediately previous hashed value H_{i-1} hold:

$$H_i = h(M_i, H_{i-1}) = E_{M_i}(H_{i-1}) \oplus H_{i-1}$$

where the symbol \oplus denotes modulo-2 addition and the symbol H_i is an accumulated sum of hashed values and a message at a time i from the beginning of the document to a time $i - 1$.

Definition 1 (Multiple Davies-Meyer function). An m -bit block cipher which uses an am -bit key K which satisfies $a > 0$. Keys have different values from each other, so that h_1, h_2, \dots, h_n are Davies-Meyer functions which are different from each other. A multiple Davies-Meyer function affine transforms an m -bit message input and maps the affine transformed input to n pairs (X_i, Y_i) which will be used as inputs. Outputs are concatenation of h_1, h_2, \dots, h_n . At the time of a collision or a preimage, if a pair (X_i, Y_i) which forms an input block is different from the original pair, $h(X_i, Y_i)$ is active. Conversely, two functions $h(X_i, Y_i)$ and $h(X_j, Y_j)$ are independently attackable, if a variable parameter (X_j, Y_j) of the function h_j does not change despite a change in a variable parameter (X_i, Y_i) of the function h_i .

ASSUMPTION 2 [KP96,KP97]: Assume we found a collision or preimage to multiple Davies-Meyer compression functions. Consider P expresses the number of active functions and $P - v$ expresses the maximum number of independently attackable functions. At least $2^{vm/2}$ or 2^{vm} encryptions are respectively necessary for a collision or preimage to occur.

3 Construction Method Using Error Correcting Codes

3.1 Construction Method of Knudsen L. and Preneel B.

We will now describe a construction method proposed by Knudsen L. and Preneel B. which uses error correcting codes.

Theorem 1. *Assume input blocks are encoded using (n, k, d) codes on $GF(2^{a+1})$ satisfying $(a+1)k > n$ but $a \geq 1$ and $m >> \log_2 n$. In this condition, as far as the Assumption 2 holds, at least $2^{(d-1)m/2}$ encryptions are necessary to find a collision to a compression function and at least $2^{(d-1)m}$ encryptions are necessary to find a preimage to the compression function [KP97]. This hash function requires an internal memory of nm bits, and a hash rate is $(a+1)(k/n) - 1$.*

That is, if we use error correcting codes having a distance of 3, we can improve the security level for collision attacks from $2^{m/2}$ to 2^m or the security level for preimage attacks from 2^m to 2^{2m} and easily construct a secure hash function.

3.2 Construction Method Using BCH Codes

Let us apply Theorem 1 to binary BCH codes. Although $a \geq 1$ is assumed, we consider binary BCH codes inserting $a = 0$. This constructs (n, k, d) codes over $GF(2)$, which leads $k > n$. This is impossible because $k < n$ is required in order to construct error correction codes. To construct error correction codes if we take two BCH codewords, then $a = 0$ has no significance, and thus we obtain the construction of $2k > n$. Therefore, modification allotting the each bit of input block, to the k elements of BCH codeword, enables the new construction. Instead of assigning symbol elements of Galois fields from two m -bit inputs which are basic structures of the Davies-Meyer method, the method requires to assign two codewords of a binary code to the inputs so as to allow use of binary codes. These are one for previous hashed values, and the other for message block, then we obtain an $n \times 2$ input array. We allot one element of one binary codeword to one bit of one m -bit block, respectively, not allotting symbols of the Galois field element.

Theorem 2 (Theorem 1 extended). *Assume input blocks are encoded using (n, k, d) codes on $GF(2)$ satisfying $k < n$ and $2k > n$, and $m >> \log_2 n$. In this condition, as far as the Assumption 2 holds, at least $2^{(d-1)m/2}$ encryptions are necessary to find a collision to a compression function and at least $2^{(d-1)m}$ encryptions are necessary to find a preimage to the compression function [KP97]. This hash function requires an internal memory of nm bits, and a hash rate is $(a+1)(k/n) - 1$.*

(Proof): At least d Davies-Meyer functions are active and also at least first k bits among n bits of functions h_i are independent each other. At least $d - 1$ bits among the last $n - k$ bits are dependent on the first k input. The condition of $n - k \geq d - 1$ gives Theorem 2 naturally. **(QED).**

However, binary codes are not so efficient than non-binary codes are, because there is no non-trivial MDS codes for binary codes, therefore the hash rate becomes very low. We can construct using two $(7,4,3)$ Hamming codes in order to improve collision resistance, for example, from $2^{m/2}$ to 2^m , although hash rate is down to $1/7$. We devise efficiency by error correction encoding only message vectors.

4 Our Proposed Methods

A BCH codeword in the first column which consists of n pieces of m -blocks which are constructed only by hashed values of one unit time ago, encodes only a hashed value of a previous time point, which prohibits to enter information of a new message. Hence, this column will not be attacked nor have to be encoded against attacks. Only n pieces of m -blocks in the second column must be encoded and protected against attacks. We propose, in this paper again, not to encode the first column which consists of blocks of one unit time ago alone which are fed back but to encode only the second column. This realizes an efficient construction which allows to input more new messages instead.

4.1 Construction Method Using Block Codes

Now, let's assume a collision or preimage has occurred to n concatenated m -bit hashed values.

Theorem 3. *Assume error correction encoders to encode k message blocks into n m -bit blocks, then input n previous hashed values and error correction codeword making $n \times 2$ m -bit blocks to n multiple Davies-Mayer functions. Assume a collision or preimage among n consecutive blocks occur, and if P Davies-Mayer functions are active, then $P - (d - 1)$ message inputs Y_i s are independent, but $d - 1$ message Y_i s are depend. In this condition, as far as the Assumption 2 holds, at least $2^{(d-1)m/2}$ encryptions are necessary to find a collision to a compression function and at least $2^{(d-1)m}$ encryptions are necessary to find a preimage to the compression function.*

(Proof): At least d Davies-Meyer functions are active and also at least first k bits among n bits of vector Y_i s are independent each other. At least $d - 1$ bits among the last $n - k$ bits are dependent from the first k input. The condition of $n - k \geq d - 1$ gives Theorem 3 naturally. **(QED).**

The discussion mentioned above, holds as long as the state is continuous and the input vectors X_i s are regarded as random oracles. However, the initial state

gives arbitrary values for key and message too. Therefore the initial values must be treated based on Knudsen and Preneel.

EXTENDED ASSUMPTION 2

Initial state Suppose a collision or preimage is found for multiple Davies-Meyer compression functions. Let P be the number of active functions, and $P - v$ be the number of attackable functions, and h_1, h_2, \dots, h_n be Davies-Meyer functions which collide, simultaneously. Among these functions, the relationship $h_i(X_i, Y_i) = E_{X_i}(Y_i) \oplus Y_i$ holds. These are obtained by fixing $\lceil \log_2 n \rceil$ key bits to different values. The compression functions of a multiple Davies-Meyer scheme takes $2km$ -bit input blocks, which are expanded by an affine mapping to the n pairs (X_i, Y_i) . The output of the compression functions depend on all $2k$ input blocks, thus, the matrix of the affine mapping has the rank $2k$.

Steady state In the steady state, except initial state the compression function of multiple Davies- Meyer scheme take km -bit input blocks, which are expanded by an affine mapping to the pairs (X_i, Y_i) . The output of the compression functions depend on all k input blocks, therefore the matrix of the affine mapping has the rank k . Suppose a collision or preimage for the compression function of a multiple Davies-Meyer scheme is found, simultaneously for h_1, h_2, \dots, h_n . Assume the two different inputs of $\{Z_i\}$ and $\{Z'_i\}$ give all the same outputs of n blocks. Let P functions group be $\{h_j\}$ which $Z_i \neq Z'_i$ holds under the condition of $P \leq n$. The matrix of functions h_j has the rank $P - v$.

The simultaneous collision requires $2^{mv/2}$ encryptions and the simultaneous preimage requires 2^{mv} encryptions concerning to P functions h_j . Consequently, the error-correction encoding of the each k bits within nm -bit blocks, gives n bits of initial value X_0 , and the remaining $n - k$ bits are allotted in initial input vector Y_0 .

Namely, the initial message bits are $r = 2k - n$ bits. Except initial condition, message bits are given by $r = k$. As a result, $d - 1$ vectors Y_j at the last $n - k$ vectors are required to depend on the first k vectors Y_j s, so in the steady state, only the vector Y_j s are required error correction encoding.

Corollary 1. *Let us consider (n, k, d) binary code. The previous hashed values H_{i-1} s at a time $i - 1$ are fed back to nm -bit blocks of the first column at a time i . The number of message blocks becomes k . The hash rate is given by k/n .*

This rate is the same rate as a code rate that an error correction code originally has. Therefore hash rate is greatly improved. As operational notice nm bit initial key values are required. In the new construction method, one codewords of binary codes allows to input to m -bit blocks which are one parts of n input pairs. The construction will be now described using $(7, 4, 3)$ Hamming codes for the simplicity of description. A parity matrix is as follows:

$$H = (\alpha^6, \alpha^5, \alpha^4, \alpha^3, \alpha^2, \alpha^1, \alpha^0)$$

$$\begin{array}{r}
 1\ 1\ 1\ 0\ 1\ 0\ 0 \\
 = 0\ 1\ 1\ 1\ 0\ 1\ 0 \\
 1\ 1\ 0\ 1\ 0\ 0\ 1
 \end{array}$$

A primitive polynomial is $X^3 + X + 1$.

$$H_1 = h_1(G_1, M_1)$$

$$H_2 = h_2(G_2, M_2)$$

$$H_3 = h_3(G_3, M_3)$$

$$H_4 = h_4(G_4, M_4)$$

$$H_5 = h_5(G_5, r_3)$$

$$H_6 = h_6(G_6, r_2)$$

$$H_7 = h_7(G_7, r_1),$$

where the H_i 's are hash values denoted by eq. (1) and the symbols r_3, r_2 and r_1 are blocks of check bits of Hamming codes. The blocks r_3, r_2 and r_1 are expressed as:

$$r_3 = M_1 \oplus M_2 \oplus M_3$$

$$r_2 = M_2 \oplus M_3 \oplus M_4$$

$$r_1 = M_1 \oplus M_2 \oplus M_4$$

where the symbols G_i, M_i and r_i denote m -bit blocks. The m -bit block G_i expresses a hashed value at a previous time point. The m -bit block M_i is a message block. The m -bit block r_i expresses the check bits of the Hamming codes. Although we have used Hamming codes for the convenience of description, let's now consider double error correcting BCH codes for the sake of the larger distance. Use of double error correcting BCH codes makes it possible to construct the more secure hash function. Assume (31, 21, 5) BCH codes are shortened to (30, 20, 5) BCH codes. Since $r = 20$, a hash rate is $20/30 = 2/3$. For comparison, Table 1 shows results of an example where binary codes are used and also an example according to Knudsen L. and Preneel B.

As described above, according to this paper, use of binary codes omits computation of Galois fields and allows to obtain a hashed value which is highly secure.

4.2 Construction Method Using Convolutional Codes

When we use block codes, as a tendency, a code length becomes longer to increase the number of error correcting capability and computation accordingly becomes complex. In addition, since many Davies-Meyer units are necessary because of a long code length, it is necessary to prepare a number of apparatuses for realizing Davies-Meyer scheme. We will now introduce a construction method which

Table 1. A method using binary codes vs. a conventional method

	field	t	code	rate	collision	memory
ours	$GF(2)$	1	(7,4,3)	$4/7=0.571$	2^m	7m
	$GF(2)$	1	(15,11,4)	$11/15=0.733$	2^m	15m
	$GF(2)$	2	(15,7,5)	$7/15=0.467$	2^{2m}	15m
	$GF(2)$	2	(25,15,5)	$15/25=0.60$	2^{2m}	25m
	$GF(2)$	2	(30,20,5)	$20/30=0.666$	2^{2m}	30m
	$GF(2)$	2	(62,54,5)	$54/62=0.871$	2^{2m}	62m
Knudsen & Preneel	$GF(2^2)$	1	(5,3,3)	$1/5=0.20$	2^m	5m
	$GF(2^4)$	1	(6,4,3)	$1/4=0.25$	2^m	6m
	$GF(2^2)$	1	(8,5,3)	$1/4=0.25$	2^m	8m

requires to select the number of multiple Davies-Meyer functions, to the same as the sub-block length n_0 of a convolutional code and thereby reduce the size of a Davies-Meyer function down to the sub-block length n_0 of a convolutional code from a code length n which is a code length in a case where a block code is used. In short, the number of units of the basic structure of the Davies-Meyer function is reduced to $1/N$.

Theorem 4. *Encode k_0 message blocks each into n_0 convolutional subcode words and construct $n_0 \times 2$ inputs for Multiple Davies-Meyer scheme with n_0 previous hashed value and n_0 convolutional subcode words. Let the constraint length of the convolutional codes be N . A collision or a preimage happens on $N \times n_0$ consecutive m -bit blocks, and P Davies-Meyer functions are active, then $P - (d - 1)$ input vectors Y_j 's are independent and remaining $d - 1$ inputs depend on the first k inputs. In this condition, as far as the Assumption 2 holds, at least $2^{(d-1)m/2}$ encryptions are necessary to find a collision to a compression function and at least $2^{(d-1)m}$ encryptions are necessary to find a preimage to the compression function.*

As an operational notice, the initial key value of $n_0 m$ bits are required. Our construction is characterized by a Convolutional encoder, a multiple Davies-Meyer function, and an FIFO memory which accumulates hashed values which are concatenated to N time units. Roughly speaking, there are two types of Convolutional encoders for Convolutional codes. That is, type 1 encoders and type 2 encoders introduced by Massey J.L.[Mas63]. This will be described with reference to an example. Convolutional codes with sub-block length n_0 bits, sub-information symbol bits k_0 , distance d and constraint length N which is a span of N time units, are called $(n_0, k_0, d; N)$ Convolutional codes. Consider a type 1 encoder for $(3, 2, 3; 3)$ Convolutional codes. Two sub-generators are expressed as follows: [Lin70]

$$g(1, 1) = 101, g(2, 1) = 110$$

The respective sub-generator elements are:

$$g_0(1, 1) = 1, g_1(1, 1) = 0, g_2(1, 1) = 1$$

$$g_0(2, 1) = 1, g_1(2, 1) = 1, g_2(2, 1) = 0.$$

Constructed with these, an encoder is obtained [Mas63]. A type 2 construction of an encoder according to Massey J.L. is also available using this method [Mas63]. Let us consider an example of operations of a construction which uses the type 1 construction of Massey J.L. and the encoder for (3, 2, 3; 3) Convolutional codes. The symbols M_{1i} and M_{2i} denote messages which are m -bit blocks, the symbol H_i denotes $n_0 m$ -bit block, and the symbol C_i denotes an m -bit block of a check of a Convolutional code. H_j holds the relation $H_j = H_i$, in which case a delay circuit of one unit time ago is not necessary if a multiple Davies-Meyer function is of a D-latch input type. If the multiple Davies-Meyer function is constructed with wired logic, $H_j = H_{i-1}$ holds. On the other hand, C_{i-2} and M_i are inputted at the same time, since the input side of the multiple Davies-Meyer function is C_i . This applies to the following as well. H_0 ($i = 0$ for H_i) is supplied as an initial value to the input side of the multiple Davies-Meyer function, and initial values C_{10} and C_{20} are supplied respectively to check symbol registers C_1 and C_2 . (Suppose C_1 is closer to multiple Davies-Meyer and C_1, C_2 are cascade connected). First messages M_{1i} and M_{2i} are added to each other by modulo-2 addition and input as a check symbol C_i . A hashed value H_i of $3 \times 64 = 192$ bits is obtained as a result of one operation of the multiple Davies-Meyer function and supplied to the FIFO memory while at the same time fed back to the input side. The next set of a message, a check symbol and a hashed values is then input and a hashed value H_2 is obtained as a result of the next operation. H_i are generated one after another in this manner, so that the FIFO memory always stores a hashed value $H_{i-2} \parallel H_{i-1} \parallel H_i$ which is equivalent to $3 \times 3 = 9$ pieces of m -bit blocks in total. Since Convolutional encoding is encoded after data are all input, $N - 1$, i.e., $2 \times 2 = 4$ pieces of m -bit blocks with a data value 0 are input and computation of a hashed value completes.

Although the size of a multiple Davies-Meyer function is reduced down to $1/N$ if Convolutional codes are used, in order to obtain a hashed value having a constraint length N and $N \times m$ bits, it is necessary to input 0 of $(N - 1)m$ bits at the end of data so that the output data will be taken completely from an encoder. This causes a delay which is $(N - 1)$ times as large as a unit time of encoding. In the construction according to this method, $n_0 \times 2$ pieces of two-dimensional arrangements are constructed on the input side of the multiple Davies-Meyer function, n_0 hashed values of one unit time ago are fed back to n_0 pieces of m -bit blocks in the first column, k_0 pieces of m -bit blocks of a new message are encoded into n_0 pieces of m -bit blocks and thereafter input. Table 2 shows the number of dummy data which are needed at initialization and termination of the construction according to this paper. Table 3 shows an example where Wyner-Ash codes are used and Table 4 shows an example where CSOC codes (Convolutional Self-Orthogonal Codes) are used, respectively [Lin70].

Table 2. Dummy data for the proposed construction

	initial	final
hash dummy (any character)	n_0 (m -bit block)	
hash dummy(0)		$(N - 1)k_0$ (m -bit block)
check dummy (0)	$N - 1$ (m -bit block)	

Table 3. A construction from Wyner-Ash codes

Code (W-A) ($n_0, k_0, d; N$)	hash rate	collision	memory(hash)
(4,3,3;2)	3/4	2^m	8m
(8,7,3;2)	7/8	2^m	16m

5 Summary

We developed the method which uses binary codes and the method which uses convolutional codes from the Knudsen and Preneel's method which requires to construct a hash function using error correcting codes, and described in this paper constructions according to our methods. In recent years, we have seen serious discussions on the security of key length of block ciphers, and ciphers with 128 bits or more are becoming a main stream these days. To respond to such current demands, researches for combining conventional methods to obtain secure hash functions should be more and more actively conducted.

In relation to the selection process which is ongoing for AES (Advanced Encryption Standard) in U.S., it is necessary to establish a method of constructing a hash function using a block cipher which has a longer key length. At the same time, continued researches on specific algorithms for hash functions are necessary.

A challenge from now is a block cipher, such as MISTY [Mat96] and IDEA [LM90], whose encryption/decryption key is 128 bits, i.e., $a = 2$ despite an input satisfying $a \geq 2$, that is, $m = 64$ and an output of 64 bits. Meanwhile, according to tandem D-M (Tandem Davies-Meyer)[Sch96] and abreast D-M (Tandem Davies-Meyer)[Sch96], $2m$ -bit hashed value is obtained in response to an m -bit key. Inputs to a hash function are the $2m$ -bit hashed value at a time $i - 1$ and the m -bit key input. Further, generally considering a cipher which uses a key having a key length of am bits satisfying $a \geq 2$ and for which a hashed value of bm bits is obtained, the direction of rows is $a + b$. A technique for synthesizing a hash function which is applicable to such a cryptographic method should be researched. Those researches are expected to considerably improve security.

Table 4. A construction from SCOC codes

code	hash rate	collision	memory
(CSOC) $(n_0, k_0, d; N)$			
(3,2,3;3)	2/3	2^m	9m
(3,2,5;14)	2/3	2^{2m}	42m

Acknowledgments

The first author would like to appreciate Prof. Shigeichi Hirasawa (Waseda University) and Prof. Takakazu Satoh (Saitama University) for useful discussions.

References

- Dam89. Damgård, I.,B., "A design principle for hash functions," Advances in Cryptology, Proc. Crypto'89, LNCS 435, Brassard, B. Ed., Springer-Verlag, 1990, pp.416-427. [394](#)
- BB94. den Boer,B. and Bosselaers,"Collisions for the compression function of MD5," Advances in Cryptology, Proc. Eurocrypt'93, LNCS 765, T. Helleseth, Ed., Springer-Verlag, 1994, pp.293- 304.
- DP84. Davies D.W. and Price W.L., "Digital Signature An Update: Proceedings of International Conference on Computer Communications", Sydney, Oct 1984, North Holland: Elsevier, 1985, pp.843-847. [393](#)
- Dob96a. Dobbertin H., "Cryptanalysis of MD5 compress", Presented at the rump session of EUROCRYPT'96, May 1996. [391](#)
- Dob96b. Dobbertin,H. "Cryptanalysis of MD4," Fast Software Encryption, LNCS 1039, Gollman, D. Ed., Springer-Verlag, 1996, pp.53-69. [391](#)
- HLMW94. Hohl,W., Lai X., Meier T. and Waldvogel C., "Security of iterated hash functions based on block ciphers," Advances in Cryptology, Proc., Crypto'93, LNCS 773, Stinson,D., Ed., Springer-Verlag, 1994, pp.379-390.
- ISO. ISO/IEC 10118, "Information technology, Security techniques, Hash-functions, Part1: General and Part2:Hash-functions using an n-bit block cipher algorithm," .
- Knu93. Knudsen L.R., "Analysis and design of cryptographic hash functions," Doctoral Dissertation, Katholieke Universiteit Leuven, 1993.
- Knu92. Knudsen L.R., Govaerts R. and Vandewalle J., "On the power of memory in the design of collision resistant hash functions," Advances in cryptology, Proc. Auscrypt'92, LNCS 718, Seberry J. and Zheng Y., ed., Springer-Verlag, 1993, pp.105-121.
- Knu94. Knudsen L.R., "A Key-schedule Weakness in SAFER K-64," Advances in Cryptology, Proc. Crypto'94, LNCS 839, Desmedt. Y. ed., Springer-Verlag, 1994, pp.274-286.
- KL94. Knudsen L. R., and Lai X., "New attacks on all double block length hash functions of hash rate 1, including the pararell-DM," Advances in Cryptology, Proc. Euro-crypto'94, LNCS 959, De Santis, A., Ed., Springer-Verlag,1995, pp.410-418.

- KP96. Knudsen L.R.,and Preneel B., " Hash functions based on block ciphers and quaternary codes", Advances in Cryptology, Proc. Asiacrypto'96, LNCS 1163, K. Kim, T. Matsumoto, Eds., Springer-Verlag, 1996, pp.77-90. [391](#), [392](#), [394](#), [395](#)
- KP97. Knudsen L. and Preneel B. "Fast and secure hashing based on codes," Crypto'97, LNCS1294, pp.485-498, 1997. [391](#), [391](#), [392](#), [394](#), [394](#), [394](#), [395](#), [395](#)
- Lai92. Lai X.,"On the Design and Security of Block Ciphers," ETH Series in Information Processing, Vol.1, Massey J. L. Ed., Hartung-Gorre Verlag, Konstanz, 1992.
- LM90. Lai X. and Massey J., "A Proposal for a New Block Encryption Standard", Advances in Cryptology-EUROCRYPT'90 Proceedings, Springer-Verlag, 1991,pp.389-404.
- Lin70. Lin, S, "An Introduction to Error Correction Codes", Englewood Cliffs., N., J., 1970, Chapter 10.
- MS78. Macwilliams F.J. and Sloane N.J.A., "The Theory of Error-Correcting Codes," North-Holland Publishing Company, Amsterdam, 1978. [401](#) [399](#), [400](#)
- MMO85. Matyas S. M., Meyer C. H. and Oseas,J.,"Generating strong one-way functions with cryptographic algorithm," IBM Techn. Disclosure, Bull., Vol. 27, No. 10A, 1985, pp.5658-5659. [393](#)
- Me89. Merkle R.,"One way hash functions and DES," Advances in Cryptology, Proc., Crypto'89, LNCS 435, Brassard G. Ed., Spring-Verlag, 1990, pp.428-446. [394](#)
- MS88. Meyer C. H. and Schilling M.,"Secure program load with Manipulation Detection Code," Proc. Securicom 1988, pp.111-130.
- Mas63. Massey, J.L, "Threshold Decoding", MIT Press, 1963. [399](#), [400](#), [400](#)
- Mat96. Matsui M.,"New structure of block ciphers with provable security against differential and linear cryptoanalysis", the third international workshop of fast software encryption,1996. [401](#)
- MS87. Moore j. H. and Simmons G. J.,"Cyclic structure of the DES for keys having palindromic (or antipalindromic) sequences of round keys", IEEE Trans. on Software Engineering, Vol.SE-13, No.2, 1987, pp.262-273.
- NY89. Naor M. and Yung M.,"Universal one-way hash functions and their cryptographic applications," Proc. 21st ACM Symposium on the Theory of Computing, ACM, 1989, pp.387-394.
- PGV94. Preneel B., Govaerts R. and Vandewalle J.,"Hash functions based on block ciphers: a synthetic approach," Advances in Cryptology, Proc., Crypto'93, LNCS 773, Stinson, D.,Ed., Spring- Verlag, 1994, pp.368-378.
- QD89. Quisquater, J.-J. and Delescaille J.-P.,"How easy is collision search? Application to DES," Advances in Cryptology, Proc. Eurocrypt'89, LNCS 434, Quisquater, J.-J. and Delescaille J.-P., Ed., Springer-Verlag, 1990, pp.429-434.
- Riv90. Rivest, R.L.,"The MD4 message digest algorithm," Advances in Cryptology, Proc. Crypto'90, LNCS 537.S. Vanstone, Ed., Spring-Verlag, 1991, pp.303-311.
- Riv92. Rivest, R.L.,"The MD5 message digest algorithm," Request for Comments (RFC) 1321, Internet Activities Board, Internet Privacy Task force, April 1992.

- RP95. Rijmen V. and Preneel B., "Improved characteristics for differential crypt-analysis of hash functions based on block ciphers," *Fast Software Encryption, LNCS 1008*, Preneel B., Ed., Springer-Verlag, 1995, pp.242-248.
- Sch96. Schneier B.,: *Applied cryptography*, John Wiley & Sons, Inc., New York, pp.451-452, 1996. [401](#), [401](#)
- OW94. Van Oorshot, P. C. and Wiener M. J., "Parallel collision search with application to hash functions and discrete logarithms," *Proc. 2nd ACM Conference on Computer and Communications Security*, ACM, 1994, pp.210-218.

Fast Implementation of Elliptic Curve Arithmetic in $\text{GF}(p^n)$

Chae Hoon Lim and Hyo Sun Hwang

Information and Communications Research Center, Future Systems, Inc.
372-2, Yang Jae-Dong, Seo Cho-Gu, Seoul, 137-130, KOREA
`{chlimg,hyosun}@future.co.kr`

Abstract. Elliptic curve cryptosystems have attracted much attention in recent years and one of major interests in ECC is to develop fast algorithms for field/elliptic curve arithmetic. In this paper we present various improvement techniques for field arithmetic in $\text{GF}(p^n)$ (p a prime), in particular, fast field multiplication and inversion algorithms, and provide our implementation results on Pentium II and Alpha 21164 microprocessors.

1 Introduction

Elliptic curve cryptosystems, first introduced by Koblitz [10] and Miller [19], have been investigated by many other researchers in recent years. In particular, much research has been conducted on fast algorithms and implementation techniques of elliptic curve arithmetic over various finite fields [12, 22, 24, 23, 8, 7, 2, 9].

Since elliptic curve groups can provide a higher level of security with smaller key sizes, there is increasing interest also in the industry and thus a lot of active standardization processes are going on, for example, IEEE P1363 [26], ISO/IEC CD 14883-3 and DIS 11770-3, ANSI X.9.62/9.63 [27, 28], etc. Thus we can expect that elliptic curve cryptosystems will be widely used for many security applications in the near future. In this regard, it is also expected that there will be a strong demand on efficient algorithms for fast implementation of elliptic curve cryptosystems.

An elliptic curve over $\text{GF}(2^n)$ is best suited for hardware implementations, but in software an elliptic curve over $\text{GF}(p^n)$ is more attractive since better performances can be achieved with a suitable choice of parameters [2]. This paper is devoted to devising and implementing efficient methods for speeding up field arithmetic in $\text{GF}(p^n)$. In particular, we present a fast field inversion algorithm in $\text{GF}(p^n)$, which only requires one subfield inversion and thus runs significantly (more than 2 to 4 times) faster than the extended Euclidean algorithm for most sizes of p interested to us (i.e., $|p| \approx 32$ or 64). Using the speedup techniques presented in this paper, we have implemented elliptic curve arithmetic with various choices of field parameters for $\text{GF}(p^n)$. Our implementation shows that scalar multiplication can be performed more than 5 times faster than modular exponentiation for parameters of a comparable security level.

This paper is organized as follows. Section 2 briefly summarizes elliptic curve arithmetic in $\text{GF}(p^n)$ and Section 3 describes various algorithms and speed-up

techniques for field multiplication and inversion. We then present our implementation results in Section 5 and conclude in Section 6.

2 Elliptic Curve Arithmetic in $\text{GF}(p^n)$

2.1 Elliptic Addition/Doubling in Affine Coordinates

A non-supersingular elliptic curve defined over a finite field $\text{GF}(p^n)$ is a set of points (x, y) given by the cubic equation

$$y^2 = x^3 + ax + b \quad (a, b \in \text{GF}(p^n), \quad 4a^2 + 27b^3 \neq 0),$$

together with ‘point at infinity’ as an identity element. Addition formulas in this affine coordinate system are defined as:

– Addition: $(x_2, y_2) = (x_0, y_0) + (x_1, y_1)$,

$$\begin{aligned} x_2 &= \lambda^2 - (x_0 + x_1) \\ y_2 &= \lambda(x_0 - x_2) - y_0 \end{aligned} \quad \text{where } \lambda = \frac{y_1 - y_0}{x_1 - x_0}$$

– Doubling: $(x_2, y_2) = 2(x_0, y_0)$

$$\begin{aligned} x_2 &= \lambda^2 - 2x_0 \\ y_2 &= \lambda(x_0 - x_2) - y_0 \end{aligned} \quad \text{where } \lambda = \frac{3x_0^2 + a}{2y_0}$$

2.2 Elliptic Addition/Doubling in Projective Coordinates

A big disadvantage of using affine representation of elliptic curve points is that addition/doubling requires a very expensive field inversion. There is another way to represent elliptic curve points, the so-called (weighted) projective representation, which eliminates the expensive field inversion at the cost of more field multiplications. The addition/doubling formulas described here are similar to the ones in the IEEE P1363 Draft [26].

We will use the following transformation for coordinate conversions:

$$x = \frac{X}{Z^2}, \quad y = \frac{Y}{2Z^3}.$$

So, the affine coordinates (x, y) should be transformed into the corresponding projective coordinates $(X, Y, Z) = (x, 2y, 1)$. The factor 2 in y is included here to eliminate the modular division by 2 appearing in the addition formula when using $y = \frac{Y}{Z^3}$ (see A.10.5 in [26]). This also reduces the number of field additions/subtractions required in the doubling formula. Note that the addition/subtraction time in $\text{GF}(p^n)$ is not negligible (about 15% of field multiplication on P6 and Alpha; see Sect.4).

– Addition: $(X_2, Y_2, Z_2) = (X_0, Y_0, Z_0) + (X_1, Y_1, Z_1)$

$$\begin{aligned} A &= X_0 Z_1^2 + X_1 Z_0^2, \quad B = X_0 Z_1^2 - X_1 Z_0^2, \\ C &= Y_0 Z_1^3 + Y_1 Z_0^3, \quad D = Y_0 Z_1^3 - Y_1 Z_0^3, \quad E = 2B, \\ Z_2 &= EZ_0 Z_1, \quad X_2 = D^2 - AE^2, \quad Y_2 = D(AE^2 - 2X_2) - E^2 BC. \end{aligned}$$

Special case of $Z_1 = 1$:

$$\begin{aligned} A &= X_0 + X_1 Z_0^2, \quad B = X_0 - X_1 Z_0^2, \\ C &= Y_0 + Y_1 Z_0^3, \quad D = Y_0 - Y_1 Z_0^3, \quad E = 2B, \\ Z_2 &= EZ_0, \quad X_2 = D^2 - AE^2, \quad Y_2 = D(AE^2 - 2X_2) - E^2 BC. \end{aligned}$$

– Doubling: $(X_2, Y_2, Z_2) = 2(X_0, Y_0, Z_0)$

$$\begin{aligned} A &= 3X_0^2 + aZ_0^4, \quad B = 2X_0 Y_0^2, \quad C = Y_0^4, \\ Z_2 &= Y_0 Z_0, \quad X_2 = A^2 - B, \quad Y_2 = A(B - 2X_2) - C. \end{aligned}$$

The above formulas show that elliptic addition requires 12 multiplications, 4 squarings and 9 additions in $\text{GF}(p^n)$ (8 multiplications, 3 squarings and 9 additions if $Z_1 = 1$), while elliptic doubling requires 4 multiplications, 6 squarings and 7 additions. Note however that k repeated doublings can save 2 squarings in $k - 1$ doublings by using one more chaining variable for aZ_0^4 (i.e., compute $W = aZ_0^4$ at the first doubling and then update W by $W \leftarrow CW$ in each iteration except for the final doubling) [7]. Also note that if $a = -3$, we can compute A in doubling as $A = 3(X_0 + Z_0^2)(X_0 - Z_0^2)$.

2.3 Performance Comparison

Table 1 summarizes the number of field operations required for elliptic addition and doubling in each coordinate system, where D_e and A_e respectively denote elliptic doubling and addition, and I, M, S and A respectively denote field operations of inversion, multiplication, squaring and addition. The number of squarings in elliptic addition can be reduced by 2 with the special choice of $a = -3$. So, for better performances we used $a = -3$ in all our implementations. Note that the fixed value for a does not much restrict the choice of elliptic curves, since the proportion of elliptic curves that can be rescaled to have $a = -3$ is approximately 1/2 or 1/4, depending on the residue of $p \bmod 4$, for $\text{GF}(p^n)$ (see Appendix A in [26]).

Which representation of elliptic curve points gives rise to a better performance can be determined by the speed ratio of field inversion to field multiplication (I/M). Obviously, arithmetic in projective coordinates with affine representation of precomputed points (i.e., $Z_1 = 1$) almost always outperforms arithmetic using pure projective representation (i.e., $Z_1 \neq 1$). The ratio I/M at the break-even point between performances with affine and projective ($Z_1 = 1$) representations can be shown to lie between 3.6 and 7.6, assuming that $a = -3$ and $1S = 0.8M$ [15]. For example, we have $I/M = 4.4$ for the signed window algorithm for scalar multiplication (window size=4), so it is always preferable to use projective coordinates with $Z_1 = 1$ if $I > 4.4M$.

a	EC oper.	Affine	Projective ($Z_1 \neq 1$)	Projective ($Z_1 = 1$)
random	D_e	$1I + 2M + 2S + 7A$	$4M + 4S + 7A$	$4M + 4S + 7A$
	A_e	$1I + 2M + S + 6A$	$12M + 4S + 9A$	$8M + 5S + 9A$
$a = -3$	D_e	$1I + 2M + 2S + 6A$	$4M + 4S + 9A$	$4M + 4S + 9A$
	A_e	$1I + 2M + S + 6A$	$12M + 4S + 9A$	$8M + 3S + 9A$

Table 1. The number of field operations for elliptic curve doubling and addition

2.4 Elliptic Scalar Multiplication

Elliptic scalar multiplication is to compute kP for a given point P on an elliptic curve and a random integer k (let $|k| = l$). The best known method for general elliptic scalar multiplication is the sliding window algorithm using addition/subtraction chains [12,6,23]. For this, we need to precompute and store odd multiples of P , $P_i = iP$ for odd i 's less than 2^w , which requires $2^{w-1} - 1$ elliptic additions and one elliptic doubling. Note that the precomputation should be done in affine coordinates for better performances, so we may use Montgomery's simultaneous inversion technique [5, Algorithm 10.3.4] to reduce the number of field inversions at the cost of more field multiplications (see also [7]). In this case, the cost for precomputation is given by $wI + (5 \cdot 2^{w-1} + 2w - 10)M + (2^{w-1} + 2w - 3)S$. Since the average interval between two consecutive windows is equal to 2 for an optimal signed encoding (e.g., see [6]), the total computational cost on average for computing kP using the signed window algorithm with window size w is approximately given by

$$\begin{aligned}
T_W &= (wI + (5 \cdot 2^{w-1} + 2w - 10)M + (2^{w-1} + 2w - 3)S) \\
&\quad + \left((l - w + 1)D_e + \left(\frac{l}{w+2} - 1 \right) A_e \right) \\
&= wI + \left(\frac{8l}{w+2} + 4l + 5 \cdot 2^{w-1} - 2w - 14 \right) M \\
&\quad + \left(\frac{3l}{w+2} + 4l + 2^{w-1} - 2w - 2 \right) S,
\end{aligned} \tag{1}$$

where we assumed to use projective representation with $Z_1 = 1$.

A different approach for computing kP was introduced by Montgomery, based on the observation that the x -coordinate of the sum of two points can be computed only using the x -coordinates of the two points if their difference is known [20] (see also [1,17]). Let $k = \sum_{i=0}^{l-1} k_i 2^i$ be the binary representation of the multiplier k (assume that $k_{l-1} = 1$). Montgomery's method successively updates a pair of points (x -coordinates only) S_i, T_i ($S_0 = P, T_0 = 2P$), while maintaining the invariant relationship $T_i - S_i = P$, by computing, for each k_i , $(S_{i+1} = 2S_i, T_{i+1} = S_i + T_i)$ if $k_i = 0$ and $(S_{i+1} = S_i + T_i, T_{i+1} = 2T_i)$ if $k_i = 1$ ($i = l-2, \dots, 1, 0$). Thus, we need one elliptic addition and one elliptic doubling for each bit of k , regardless of the Hamming weight of k . Each of elliptic addition and doubling can be done in 3 multiplications and 2 squarings in projective coordinates using Montgomery's alternative parameterization of elliptic curves

($By^2 = x^3 + Ax^2 + x$ for some A and B). So, the total cost for computing the x -coordinate of kP is given by

$$T_M = (l - 1)(A_e + D_e) = (l - 1)(6M + 4S). \quad (2)$$

Let us compare the performance of the signed window algorithm and Montgomery's method. With optimal window size $w = 4$ for most interesting values of l , we have $T_W = 4I + (5.33l + 18)M + (4.5l - 2)S = 4I + (8.93l + 16.4)M$ from equation (1), assuming that $1S = 0.8M$. Since $T_M = 9.2(l - 1)M$ from equation (2), we can see that the signed window algorithm is asymptotically faster than Montgomery's method. For a typical value of $l = 160$, we have $T_W = 4I + 1445M$ and $T_M = 1463M$, so Montgomery's method may be a little bit faster. However, Montgomery's method is not a general algorithm for elliptic scalar multiplication in $\text{GF}(p^n)$, since it can't compute the y -coordinate of kP (note however that this is not the case in $\text{GF}(2^n)$; see [17]). Of course, this may not be a problem in many applications, since most elliptic curve variants of key exchange and digital signature schemes only make use of x coordinates (e.g., see [26, 27, 28]). We thus use the signed window algorithm for scalar multiplication in this paper. Finally, it is worth noting that there are several advantages in Montgomery's method: it does not require precomputation, which may be desired for implementations in a limited computing environment (e.g., an implementation on low-cost smart cards), addition and doubling can be performed in parallel on multi-processor architectures or in hardware implementation, and the execution time does not depend on the Hamming weight of multipliers, which helps to prevent timing attacks.

On the other hand, we may use an elliptic curve defined over $\text{GF}(p)$ as an elliptic curve over $\text{GF}(p^n)$ (let us call such a curve as a subfield curve). A big advantage of using such a subfield curve is that elliptic scalar multiplication on a subfield curve can be substantially speeded up using Frobenius expansion [9, 15] (see also [11, 18, 23, 21, 4]). Though subfield curves allow much faster implementations and easy parameter generation, we should be careful for security concerns related to the special structure (e.g., see [14, 25]). We do not consider such a special curve in this paper, but for comparison we provide the implementation result from [15] in Sect. 4.

3 Speeding up Field Arithmetic

Optimization of field arithmetic is much more critical to the overall performance of elliptic scalar multiplication than optimization in group operations. This section describes various algorithms and techniques for speeding up field multiplication and inversion in $\text{GF}(p^n)$.

3.1 Field Construction

The performance of field arithmetic in $\text{GF}(p^n)$ ($n > 1$) heavily depends on the choice of parameters for field extension (a prime p and an irreducible polynomial

$f(x)$). For fast field arithmetic in $\text{GF}(p^n)$, Bailey and Paar [2] proposed to choose the parameters p , n and $f(x)$ such that $p = 2^m - c$ with small c and m around a target computer word size and $f(x) = x^n - \omega$ with small ω (called an Optimal Extension Field (OEF)).

For further optimization of field arithmetic, we placed another restriction on the size of p : choose p , whenever possible, so that multiplication results of two $|p|$ -bit numbers can be accumulated as many as possible without overflow in computer's word boundary. Then, with such a p one can reduce the number of reductions mod p required for field multiplication from n^2 to n . This will result in a substantial improvement in the overall performance, since modular reduction is still quite expensive even with the special choice of p in typical microprocessors, such as P6 and Alpha.

The field parameters selected for use in our implementations are summarized in Table 2. Of course, there are other possible choices worth considering, such as $(p = 2^{31} - 1, f(x) = x^6 - 5)$, $(p = 2^{61} - 1, f(x) = x^3 - 37)$, $(p = 2^{84} - 35$ or $2^{85} - 19, f(x) = x^2 - 2)$ and $p = 2^{166} - 5$. The three field parameters with degree of n^* ($n = 7, 11, 13$) were included for use in building subfield curves. The figures of the 'order' column in Table 2 denote the largest possible prime orders in $E/\text{GF}(p^n)$.

n	order (bits)	$p = 2^m - c$	$f(x)$
13*	168	$2^{14} - 3$	$x^{13} - 2$
12	168	$2^{14} - 3$	$x^{12} - 2$
11*	160	$2^{16} - 437$	$x^{11} - 2$
10	160	$2^{16} - 165$	$x^{10} - 2$
7*	168	$2^{28} - 57$	$x^7 - 2$
6	168	$2^{28} - 165$	$x^6 - 2$
5	160	$2^{32} - 5$	$x^5 - 2$
3	171	$2^{57} - 13$	$x^3 - 2$
2	178	$2^{89} - 1$	$x^2 - 3$
1	160	$p = 2^{160} - 2933$	

Table 2. Selected parameters for extension field $\text{GF}(p^n)$

3.2 Field Multiplication and Squaring

Let $A(x)$ and $B(x)$ be polynomials of degree $n - 1$ over $\text{GF}(p)$, i.e.,

$$A(x) = \sum_{i=0}^{n-1} A_i x^i, \quad B(x) = \sum_{i=0}^{n-1} B_i x^i,$$

where $A_i, B_i \in \text{GF}(p)$. Field multiplication in $\text{GF}(p^n)$ is to compute $C(x) = A(x)B(x) \bmod f(x)$.

First, consider methods for speeding up polynomial multiplication. There exists a well-known divide-and-conquer technique, called the Karatsuba-Ofman algorithm, to reduce the number of subfield multiplications required for polynomial multiplication [13, Sect.4.3.3]. We can use Karatsuba-Ofman's algorithm in two directions. For example, for $n = 6$, let $A(x) = A_h(x)x^3 + A_l(x)$, $B(x) = B_h(x)x^3 + B_l(x)$, where

$$\begin{aligned} A_h(x) &= A_5x^2 + A_4x + A_3, & A_l(x) &= A_2x^2 + A_1x + A_0, \\ B_h(x) &= B_5x^2 + B_4x + B_3, & B_l(x) &= B_2x^2 + B_1x + B_0. \end{aligned}$$

Then we can compute $C(x) = A(x)B(x)$ as $C(x) = D_h(x)x^6 + D_m(x)x^3 + D_l(x)$, where

$$\begin{aligned} D_h(x) &= A_h(x)B_h(x), & D_l(x) &= A_l(x)B_l(x), \\ D_m(x) &= (A_h(x) + A_l(x))(B_h(x) + B_l(x)) - (D_h(x) + D_l(x)). \end{aligned}$$

This high-level application of Karatsuba-Ofman's algorithm reduces the number of subfield multiplications from $n^2 = 36$ to $\frac{3}{4}n^2 = 27$ at the cost of more subfield additions. Our implementation shows that this technique is only effective on Alpha 21164 for n up to 7.

We may apply Karatsuba-Ofman's algorithm once again to polynomial multiplication of degree 3. However, for small n , it is better to use Karatsuba-Ofman's algorithm at the lowest word level. For example, for $n = 3$, $C(x) = (A_0 + A_1x + A_2x^2)(B_0 + B_1x + B_2x^2)$ can be computed as

$$C(x) = D_0 + (D_3 - D_0 - D_1)x + (D_4 + D_1 - D_2 - D_0)x^2 + (D_5 - D_2 - D_1)x^3 + D_2x^4,$$

where

$$\begin{aligned} D_0 &= A_0B_0, & D_1 &= A_1B_1, & D_2 &= A_2B_2, & D_3 &= (A_0 + A_1)(B_0 + B_1), \\ D_4 &= (A_0 + A_2)(B_0 + B_2), & D_5 &= (A_1 + A_2)(B_1 + B_2). \end{aligned}$$

We can thus reduce the number of subfield multiplications from 9 to 6 (reduction rate of $2/3$). Of course, this low-level Karatsuba-Ofman's algorithm can be applied to polynomial multiplication of any degree n . In this case, it is easy to see that the number of subfield multiplications required can be reduced to $\frac{n(n+1)}{2}$. However, as n becomes larger, the increase in the addition (and memory access) complexity may be larger than the reduction in the multiplication complexity. So, the effectiveness of this method varies from architecture to architecture, depending on the relative speed of basic operations involved and on the number of general-purpose registers available. For example, our implementation shows that this technique is only useful for $n = 2$ or 3 on Pentium II and for n up to 7 on Alpha 21164.

On the other hand, we can reduce the number of reductions mod p by accumulating individual product terms, A_iB_j 's, as many as possible, and then reducing the accumulated sum mod p only once. To see this, let us express

$C(x) = A(x)B(x) \bmod f(x)$, using the identity $x^n = \omega \bmod f(x)$, as

$$\begin{aligned} C(x) &= \sum_{i=0}^{n-1} A_i x^i \cdot \sum_{j=0}^{n-1} B_j x^j \bmod f(x) \\ &= \sum_{i+j=k < n} A_i B_j x^k + \omega \sum_{i+j=k \geq n} A_i B_j x^{k-n} \\ &= \sum_{k=0}^{n-1} \left(\sum_{i=0}^k A_i B_{k-i} + \omega \sum_{i=k+1}^{n-1} A_i B_{n+k-i} \right) x^k, \end{aligned}$$

so the coefficient C_k can be computed by

$$C_k = \left(\sum_{i=0}^k A_i B_{k-i} + \omega \sum_{i=k+1}^{n-1} A_i B_{n+k-i} \right) \bmod p. \quad (3)$$

Therefore, with this accumulation-and-then-reduction technique, we can reduce the number of reductions mod p from n^2 to n . We still need n^2 multiplications of $|p|$ -bit integers, assuming ω is very small (typically 2 or 3), so that multiplication by ω can be done by a few additions (this is possible in all our field constructions, as can be seen in Table 2). However, our experiments on P6 and Alpha show that modular reduction is more expensive than multiplication in most interesting fields (of course, except for large p). So, the above accumulation-and-then-reduction technique actually contributes to the overall performance more than any other optimization technique.

Since C_k is at most $2m + \lceil \log_2(\omega(n-1) + 1) \rceil$ bits long, we can do modular reduction using at most 2 multiplications by c , as long as $\lceil \log_2(\omega(n-1) + 1) \rceil + 2|c| \leq m$, which is the case for most choices of p and $f(x)$. Equation (3) suggests that it is preferable to choose p such that the largest partial product sum, C_0 , do not produce an additional carry in word boundary of a target computer, as in the parameters given in Table 2. For example, we chose $p = 2^{28} - 165$ and $f(x) = x^6 - 2$ for $\text{GF}(p^6)$, instead of $p = 2^{31} - 1$ and $f(x) = x^6 - 5$.

Finally, note that field squaring in $\text{GF}(p^n)$ only requires $\frac{n(n+1)}{2}$ multiplications of $|p|$ -bit numbers, while the number of modular reductions remains the same. Thus, though all optimization techniques described above can be applied equally well to field squaring, depending on p field squaring may not be improved as much as expected, compared to field multiplication.

3.3 Field Inversion

Field inversion in $\text{GF}(p^n)$ ($n > 1$) corresponds to computing a multiplicative inverse of a polynomial modulo an irreducible polynomial $f(x)$ of degree n . Inversion in $\text{GF}(2^n)$ can be best performed by the *almost inverse algorithm* (AIA) [22]. However, the AIA is not effective at all for polynomial inversion in $\text{GF}(p^n)$. Rather, the extended Euclidean algorithm runs a little bit faster. In this section we present fast algorithms for polynomial inversion in $\text{GF}(p^n)$ and compare their computational complexity with other known methods in [9,3].

Variant of Extended Euclidean Algorithm We start with a polynomial version of Extended Euclidean algorithm shown in Table 3 and improves it in step-by-step. Let $\deg(A)$ be the degree of $A(x)$.

Algorithm IE
Input: $A(x)$ and $f(x)$ such that $\deg(A) < n$ and $\deg(f) = n$. Output: $B(x)$ such that $A(x)B(x) = 1 \bmod f(x)$.
1. set $B \leftarrow 0$, $C \leftarrow 1$, $F \leftarrow f(x)$ and $G \leftarrow A(x)$. 2. repeat the following steps while $\deg(F) \neq 0$: 2-1. if $\deg(F) < \deg(G)$, then exchange F, B with G, C , respectively. 2-2. update F and B as follows (let $j = \deg(F) - \deg(G)$): $\alpha \leftarrow F_{\deg(F)}G_{\deg(G)}^{-1}$, $F \leftarrow F - \alpha x^j G$, $B \leftarrow B - \alpha x^j C$. 3. return $B \leftarrow F_0^{-1}B$.

Table 3. Polynomial version of Extended Euclidean algorithm

Algorithm IE reduces the degree of the larger out of $F(x)$ and $G(x)$ by at least one in each iteration of step 2. Thus, we need at most $2n - 2$ iterations of step 2 in total. The most time-consuming operation in Algorithm IE is subfield inversion for most preferable choices of p , which is much slower than even a field multiplication on Pentium II. So we first try to reduce the number of subfield inversions required by using some parallelism in step 2 of Algorithm IE. The idea is to reduce the degree of $F(x)$ or $G(x)$ by two or more at a time as shown in Table 4.

Algorithm IP
Input: $A(x)$ and $f(x)$ such that $\deg(A) < n$ and $\deg(f) = n$. Output: $B(x)$ such that $A(x)B(x) = 1 \bmod f(x)$.
1. set $B \leftarrow 0$, $C \leftarrow 1$, $F \leftarrow f(x)$ and $G \leftarrow A(x)$. 2. repeat the following steps while $\deg(F) \neq 0$: 2-1. if $\deg(F) < \deg(G)$, then exchange F, B with G, C , respectively. 2-2. update F and B as follows ($j = \deg(F) - \deg(G)$): $\alpha \leftarrow F_{\deg(F)}G_{\deg(G)}^{-1}$, $\beta \leftarrow (F_{\deg(F)-1} - \alpha G_{\deg(G)-1})G_{\deg(G)}^{-1}$, $F \leftarrow F - (\alpha x^j + \beta x^{j-1})G$, $B \leftarrow B - (\alpha x^j + \beta x^{j-1})C$. 2-3. if $\deg(F) = \deg(G)$, then execute the following: $\alpha \leftarrow F_{\deg(F)}G_{\deg(G)}^{-1}$, $F \leftarrow F - \alpha G$, $B \leftarrow B - \alpha C$. 3. return $B \leftarrow F_0^{-1}B$.

Table 4. Parallel version of Algorithm IE

In Algorithm IP, each iteration of step 2 reduces the degree of $F(x)$ by at least two by subtracting a suitable multiple of $G(x)$. Thus, the numbers of

subfield inversions and modular reductions are now reduced by half, compared to Algorithm IE, though we need the same number of $|p|$ -bit multiplications. This improvement is crucial to the overall performance in actual implementations, since both subfield inversion and modular reduction are more expensive than multiplication for most sizes of p interested to us (i.e., $|p|$ around 32 or 64). Note that we don't have to compute the first two highest coefficients of $F(x)$ in step 2-2, since they must be zero. Also note that if $|\deg(F) - \deg(G)|$ is equal to 1 at the beginning of step 2, this difference is maintained throughout the iteration with probability of $1/p$. Thus, step 2-3 will not be executed in most cases.

Field Inversion Using Multiplication Since subfield inversion becomes more and more expensive as the size of p increases, a natural question to ask is how to minimize the number of subfield inversions in a field inversion algorithm. Fortunately, we are able to modify Algorithm IE/IP into algorithms requiring only one subfield inversion at the final stage. First note that Algorithm IE maintains the following relationships throughout its internal processing:

$$A(x)B(x) + U(x)f(x) = F(x), \quad A(x)C(x) + V(x)f(x) = G(x),$$

for some polynomials $U(x)$ and $V(x)$ (not interested to us). Therefore, we can see that these relations still hold even if we multiply both $F(x)$ and $B(x)$ (both $G(x)$ and $C(x)$, respectively) by the same constant. This observation shows that the following field inversion algorithm actually works, where we only describe the variant of Algorithm IP (A referee kindly pointed out that Algorithm IM can be constructed from algorithms in [13, Sect.4.6.1]):

Algorithm IM
Input: $A(x)$ and $f(x)$ such that $\deg(A) < n$ and $\deg(f) = n$.
Output: $B(x)$ such that $A(x)B(x) = 1 \bmod f(x)$.
1. set $B \leftarrow 0$, $C \leftarrow 1$, $F \leftarrow f(x)$ and $G \leftarrow A(x)$. 2. repeat the following steps while $\deg(F) \neq 0$: 2-1. if $\deg(F) < \deg(G)$, then exchange F, B with G, C , respectively. 2-2. update F and B as follows (let $j = \deg(F) - \deg(G)$): $\alpha \leftarrow G_{\deg(G)}^2, \quad \beta \leftarrow F_{\deg(F)}G_{\deg(G)},$ $\gamma \leftarrow G_{\deg(G)}F_{\deg(F)-1} - F_{\deg(F)}G_{\deg(G)-1},$ $F \leftarrow \alpha F - (\beta x^j + \gamma x^{j-1})G, \quad B \leftarrow \alpha B - (\beta x^j + \gamma x^{j-1})C.$ 2-3. if $\deg(F) = \deg(G)$, then execute the following: $F \leftarrow G_{\deg(F)}F - F_{\deg(F)}G, \quad B \leftarrow G_{\deg(F)}B - F_{\deg(F)}C.$ 3. return $B \leftarrow F_0^{-1}B$.

Table 5. Field inversion using multiplication

Algorithm IM requires just one subfield inversion at the final step. Comparing steps 2-2 in Algorithm IM and Algorithm IP, one can see that one subfield

inversion in Algorithm IP was replaced with one multiplication by constant in both $F(x)$ and $B(x)$ in Algorithm IM (or equivalently about $n|p|$ -bit multiplications). Consequently, Algorithm IM will be more efficient than Algorithm IP if subfield inversion is more expensive than $n|p|$ -bit multiplications.

To see how $F(x)$ is updated in step 2, let us suppose that the degrees of the current $F(x)$ and $G(x)$ are d and $d-1$, respectively. Then, the coefficient update in $F(x)$ and $B(x)$ in step 2 can be described by

$$\begin{aligned} F_i &\leftarrow \begin{cases} \alpha F_i - \gamma G_i \bmod p & \text{for } i = 0, \\ \alpha F_i - (\beta G_{i-1} + \gamma G_i) \bmod p & \text{for } 1 \leq i \leq d-2, \end{cases} \\ B_i &\leftarrow \begin{cases} \alpha B_i - (\beta C_{i-1} + \gamma C_i) \bmod p & \text{for } 0 \leq i \leq n-d-1, \\ \alpha B_i - \gamma C_i \bmod p & \text{for } i = n-d. \end{cases} \end{aligned}$$

If the degree of $f(x)$ is 2 or 3, then we can derive simple inversion formulas from Algorithm IM, which will be more efficient than direct execution of Algorithm IM since we can do more intelligent manual optimization.

- $\text{GF}(p^2)$: Let $A(x) = A_1x + A_0$ ($A_i \in \text{GF}(p)$) and $f(x) = x^2 - \omega$. Then, $B(x) = A(x)^{-1} \bmod f(x)$ can be computed by

$$B(x) = F_0^{-1}(A_1x - A_0), \text{ where } F_0 = \omega A_1^2 - A_0^2.$$

- $\text{GF}(p^3)$: Let $A(x) = A_2x^2 + A_1x + A_0$ ($A_i \in \text{GF}(p)$) and $f(x) = x^3 - \omega$. Then, $B(x) = A(x)^{-1} \bmod f(x)$ can be computed by

$$B(x) = A_2F_0^{-1}(T_2x^2 + T_1x + T_0), \text{ where}$$

$$T_0 = A_0^2 - \omega A_1 A_2, \quad T_1 = \omega A_2^2 - A_0 A_1, \quad T_2 = A_1^2 - A_0 A_2, \quad F_0 = T_1^2 - T_0 T_2.$$

Comparison of Inversion Algorithms To see the relative performance of the three inversion algorithms describe above, we calculated the number of basic operations required in each algorithm. The result is summarized in Table 6. Our experiments on P6 and Alpha microprocessors show that subfield inversion (not using table lookups) is the most expensive in general, and reduction mod p is more expensive than multiplication of $|p|$ -bit numbers. So, we separately counted the number of $|p|$ -bit multiplications, reductions mod p and inversions mod p . Note that if we do not differentiate modular reduction from multiplication, the number of multiplications in the table corresponds to the number of subfield multiplications.

In Table 6, we also included the computational complexity of Bailey and Paar's inversion algorithm in $\text{GF}(p^n)$. This algorithm computes $A(x)^{-1} \bmod f(x)$ as

$$A(x)^{-1} = (A(x)^r)^{-1} A(x)^{r-1} \bmod f(x), \text{ where } r = \frac{p^n - 1}{p - 1} = 1 + p + p^2 + \cdots + p^{n-1}.$$

Since $A(x)^r = A(x)A(x)^{r-1}$ is always an element in $\text{GF}(p)$, this algorithm also requires only one subfield inversion. Furthermore, due to the special form of

algorithm	#multiplications	#reductions	#inversions
IE	$2n^2 + n - 4$	$2n^2 + n - 4$	$2n - 1$
IP	$2n^2 + n - 4$	$n^2 - n - 2$	$n + 1$
IM ($f(x) = x^n - \omega$)	$3n^2 - 5$ $(3n^2 - n - 7)$	$n^2 + 4n - 6$ $(n^2 + 4n - 8)$	1 (1)
BP [3] $f(x) = x^n - \omega$	$t(n)(n^2 + 2n - 2) + 3n - 1$ $t(n) = \lfloor \log_2(n - 1) \rfloor + H_w(n - 1) - 1$	$t(n)(3n - 2) + 2n$	1

Table 6. Complexity for computing $A(x)^{-1} \bmod f(x)$ ($\deg(f(x)) = n$, $\deg(A(x)) = n - 1$)

$r-1$, $A(x)^{r-1}$ can be efficiently computed using $x^n = \omega \bmod f(x)$ and some basic properties of finite fields (for details, see [3]). The complexity of this algorithm is given by

$$(\lfloor \log_2(n - 1) \rfloor + H_w(n - 1) - 1)(n^2 + 2n - 2) + 3n - 1,$$

where $H_w(x)$ denotes the Hamming weight of x . In Table 6, we assumed that general polynomial multiplication of degree n can be done in n^2 multiplications and n modular reductions using the accumulation-then-reduction technique. Note that this algorithm is only useful for a binomial $f(x)$ (thus not general) and that its complexity is higher than Algorithm IM for any n .

There is another field inversion method using matrix inversion proposed in [9]. The explicit formula for $n = 3$ given in [9] is almost identical to that of Algorithm IM. However, this algorithm has a computational complexity of $O(n^3)$ and is not general either. Algorithm IM seems to be the best algorithm for field inversion in $\text{GF}(p^n)$. It is also general (works equally well with any $f(x)$), systematic and easy to implement (independent of a specific form of $f(x)$).

4 Implementation and Discussion

We have implemented various field and elliptic curve arithmetic using the algorithms and techniques presented in Section 3 on two typical microprocessors: Pentium II/266MHz (32-bit μ P; Windows 98, MSVC 5.0 with in-line assembly) and Alpha 21164/533MHz (64-bit μ P; Linux, gcc 2.95 with in-line assembly).

4.1 Timings for Field Arithmetic

To see the relative speed of three inversion algorithms described in Sect.3 in actual implementations, we measured their speed on Pentium II and Alpha 21164, as shown in Tables 7 and 8. The tables show that Algorithm IP runs about $25 \sim 45\%$ faster than Algorithm IE and that Algorithm IM runs about $30 \sim 60\%$ faster than Algorithm IP for $3 \leq n \leq 7$. We used a table lookup method for sub-field inversion in $\text{GF}(p^n)$ for $n > 7$, so the advantage of Algorithm IM disappears

Alg.	IE	IP	IM	IP/IE	IM/IE	IM/IP
$\text{GF}(p^{13})$	28.71	15.82	15.54	0.55	0.54	0.98
$\text{GF}(p^{12})$	24.96	14.28	13.79	0.57	0.55	0.97
$\text{GF}(p^{11})$	32.44	20.25	22.44	0.62	0.69	1.12
$\text{GF}(p^{10})$	27.64	16.79	19.18	0.61	0.69	1.14
$\text{GF}(p^7)$	35.90	21.79	12.13	0.61	0.34	0.56
$\text{GF}(p^6)$	29.16	17.54	9.48	0.60	0.33	0.54
$\text{GF}(p^5)$	25.10	15.02	7.20	0.60	0.29	0.48
$\text{GF}(p^3)$	38.64	26.14	9.69	0.68	0.25	0.37

Table 7. Speed of three field inversion algorithms on Pentium II/266MHz (in μsec)

Alg.	IE	IP	IM	IP/IE	IM/IE	IM/IP
$\text{GF}(p^{13})$	20.05	12.92	12.95	0.64	0.65	1.00
$\text{GF}(p^{12})$	17.28	11.29	11.13	0.65	0.64	0.99
$\text{GF}(p^{11})$	21.05	14.04	13.10	0.67	0.62	0.93
$\text{GF}(p^{10})$	14.47	10.76	10.06	0.74	0.70	0.93
$\text{GF}(p^7)$	15.98	9.78	6.18	0.61	0.39	0.63
$\text{GF}(p^6)$	13.14	8.30	5.16	0.63	0.39	0.62
$\text{GF}(p^5)$	11.08	7.28	5.12	0.66	0.46	0.70
$\text{GF}(p^3)$	10.23	7.29	2.82	0.71	0.28	0.39

Table 8. Speed of three field inversion algorithms on Alpha 21164/533MHz (in μsec)

in these cases. For inversion in $\text{GF}(p)$, we used a combination of the binary and integer extended Euclidean algorithms for a better performance.

Tables 9 and 10 show the speed of field arithmetic for various fields defined in Table 2. For comparison, we also included the timings for $\text{GF}(2^n)$. The ratio A/M ranges from 0.1 to 0.2 in $\text{GF}(p^n)$, so the addition time in $\text{GF}(p^n)$ is not negligible. The column S/M shows that $1S \approx 0.8M$ on Pentium II. But on Alpha 21164 we have $1S \approx 0.9M$ for $n \leq 7$, which would be due to better optimization in field multiplication by the Karatsuba-Ofman algorithm. The same argument also explains the lower ratio of S/M ($1S \approx 0.6M$) for $n \geq 10$, since in this case the number of subfield multiplications required for squaring is $n(n+1)/2$, compared to n^2 for multiplication (note that the Karatsuba-Ofman algorithm was applied (effective) only up to $n = 7$ on Alpha). The I/M ratio ranges from 7 to 9 for $n > 3$ on both microprocessors (we used best field inversion algorithms for each n ; see Tables 7 and 8). Note that for small n (e.g., $n < 7$), subfield inversion takes much more time than field multiplication (compare SF_Inv column with F_Mul column), which explains why Algorithm IM runs much faster than Algorithm IE or IP in these cases (in particular, as p increases). Field inversion in $\text{GF}(p)$ is extremely expensive compared to multiplication (e.g., about 30 to 40 times slower than modular multiplication).

Field	SF_Inv	F_Add	F_Sqr	F_Mul	F_Inv	A/M	S/M	I/M
GF(2^{162})		0.119	0.558	3.917	54.92	0.03	0.14	14.0
GF(p^{13})	0.072	0.343	1.517	2.042	15.79	0.17	0.74	7.73
GF(p^{12})	0.071	0.326	1.364	1.816	13.94	0.18	0.75	7.68
GF(p^{11})	0.072	0.310	1.966	2.530	20.54	0.12	0.78	8.12
GF(p^{10})	0.073	0.290	1.647	2.100	17.14	0.14	0.78	8.16
GF(p^7)	1.718	0.158	1.131	1.426	12.25	0.11	0.79	8.59
GF(p^6)	1.700	0.142	0.917	1.119	9.595	0.13	0.82	8.57
GF(p^5)	2.075	0.174	0.787	0.956	7.311	0.18	0.82	7.65
GF(p^3)	6.255	0.235	1.042	1.264	9.723	0.19	0.82	7.69
GF(p^2)	17.62	0.157	0.857	1.004	19.83	0.16	0.85	19.8
GF(p)		0.151	0.885	1.012	43.25	0.15	0.87	42.7

Table 9. Timings (in μ sec) for field operations on Pentium II 266MHz

Field	SF_Inv	F_Add	F_Sqr	F_Mul	F_Inv	A/M	S/M	I/M
GF(2^{162})		0.058	0.198	1.093	11.73	0.05	0.18	10.7
GF(p^{13})	0.081	0.217	0.956	1.710	12.96	0.13	0.56	7.58
GF(p^{12})	0.089	0.220	0.854	1.460	11.18	0.15	0.59	7.66
GF(p^{11})	0.097	0.185	0.967	1.483	13.05	0.12	0.65	8.80
GF(p^{10})	0.104	0.166	0.837	1.272	10.01	0.13	0.66	7.87
GF(p^7)	0.706	0.111	0.673	0.719	6.177	0.15	0.94	8.59
GF(p^6)	0.704	0.073	0.462	0.608	5.170	0.12	0.76	8.50
GF(p^5)	0.794	0.081	0.650	0.731	5.142	0.11	0.89	7.03
GF(p^3)	1.591	0.064	0.383	0.423	2.822	0.15	0.91	6.67
GF(p^2)	5.354	0.063	0.541	0.582	6.173	0.11	0.93	10.6
GF(p)		0.072	0.512	0.597	18.97	0.12	0.86	31.8

Table 10. Timings (in μ sec) for field operations on Alpha 533MHz

4.2 Timings for Elliptic Curve Arithmetic

Tables 11 and 12 show the speed of elliptic curve operations on Pentium II and Alpha 21164 microprocessors, respectively. The tables show that it is always preferable to use projective coordinates with affine precomputation as expected from the I/M ratios in Tables 9 and 10. Obviously, we have the best performance on Pentium II with an elliptic curve over $GF(p^5)$ and on Alpha 21164 with an elliptic curve over $GF(p^3)$. It is also worth noting that an elliptic curve in $GF(p)$ gives an almost comparable speed to the best in both microprocessors. This is due to the special choice of p such that $p = 2^{160} - 2933$. Also note that the speed ratio of elliptic doubling to addition is around 0.5 in $GF(2^n)$ and around 0.75 in $GF(p^n)$.

For comparison, we summarized the timings for elliptic scalar multiplication using Frobenius expansion in Table 13 (from [15]). The table shows that we can

coordinates		Affine			Proj. ($Z_1 = 1$)		
Field	$ k $	$P + P$	$P + Q$	kP	$P + P$	$P + Q$	kP
$GF(2^{162})$	162	63.8	64.2	12147	16.8	34.9	3978
$GF(p^{13})$	178	25.8	22.9	5034	16.7	23.2	3528
$GF(p^{12})$	178	22.9	20.4	4530	15.0	20.9	3206
$GF(p^{11})$	160	32.1	29.1	5991	20.2	27.8	4104
$GF(p^{10})$	160	27.0	24.3	5075	17.0	22.0	3446
$GF(p^7)$	168	19.2	17.4	3780	11.6	16.0	2457
$GF(p^6)$	168	15.3	13.8	3016	9.52	13.0	1997
$GF(p^5)$	160	12.5	10.9	2341	8.53	11.6	1693
$GF(p^3)$	171	16.6	14.7	3316	11.2	15.0	2397
$GF(p^2)$	178	25.3	23.7	5285	9.15	12.3	2070
$GF(p)$	160	49.2	47.4	9137	9.04	12.1	1983

Table 11. Timings (in μ sec) for elliptic curve operations on Pentium II 266MHz

achieve about 2 to 3 times speed-up with subfield curves. However, when using such a special curve, we should be careful for the security consequence related to the special structure (e.g., see [25]).

5 Conclusion

We presented various speed-up techniques for field arithmetic in $GF(p^n)$. The main improvements presented in this paper consist of various optimizations in field multiplication and inversion and careful choices of field parameters to speed up field arithmetic. Since the presented optimization techniques are focused on more primitive field arithmetic, the performance improvement in practical implementations will be more substantial than with optimizations in elliptic curve arithmetic. We also presented implementation results on two popular microprocessors, Pentium II and Alpha 21164.

Acknowledgement: The authors would like to thank an anonymous referee for his constructive comments, which were very helpful in improving the presentation of this paper.

References

1. G.B.Agnew, R.C.Mullin and S.A.Vanstone, An implementation of elliptic curve cryptosystems over $F_{2^{155}}$, *IEEE J. Selected Areas in Commun.*, 11, 5, 1993, pp.804-813. [408](#)
2. D.V.Bailey and C.Paar, Optimal extension field for fast arithmetic in public key algorithms, *Advances in Cryptology-CRYPTO'98*, LNCS 1462, Springer-Verlag, 1998, pp.472-485. [405, 410](#)

coordinates		Affine			Proj.($Z_1 = 1$)		
Field	$ k $	$P + P$	$P + Q$	kP	$P + P$	$P + Q$	kP
GF(2^{162})	162	15.0	14.9	2875	5.46	10.9	1253
GF(p^{13})	178	20.4	18.4	3970	12.1	17.7	2514
GF(p^{12})	178	18.0	16.0	3492	10.4	15.3	2204
GF(p^{11})	160	20.1	18.3	3733	11.1	16.2	2270
GF(p^{10})	160	16.6	14.9	3046	9.89	14.3	2002
GF(p^7)	168	10.4	9.13	2028	6.60	8.81	1364
GF(p^6)	168	8.37	7.48	1675	4.81	6.54	1008
GF(p^5)	160	9.01	7.86	1657	5.97	8.08	1204
GF(p^3)	171	5.64	4.87	1143	3.83	5.06	816
GF(p^2)	178	9.66	8.89	2009	5.31	6.95	1138
GF(p)	160	22.7	21.8	4216	5.42	7.34	1142

Table 12. Timings (in μ sec) for elliptic curve operations on Alpha 533MHz

μ P	algorithm		signed binary		Alg. LL		Alg. LL-SI	
	field	$ k $	A	P	A	P	A	P
Pentium II 266MHz	GF(p^{13})	178	1.96	1.85	1.66	1.57	1.46	1.36
	GF(p^{11})	160	2.34	2.11	2.01	1.83	1.77	1.60
	GF(p')	168	1.71	1.44	1.49	1.25	1.40	1.17
Alpha 21164 533MHz	GF(p^{13})	178	1.57	1.40	1.35	1.19	1.15	1.04
	GF(p^{11})	160	1.50	1.25	1.28	1.08	1.11	0.97
	GF(p')	168	0.93	0.80	0.80	0.69	0.76	0.66

Table 13. Timings (in msec) for scalar multiplication kP using Frobenius expansion

3. D.V.Bailey and C.Paar, Elliptic curve cryptosystems over large characteristic extension fields, preprint, 1999. [412](#), [416](#)
4. J.H.Cheon, S.M.Park, S.W.Park and D.H.Kim, Two efficient algorithms for arithmetic of elliptic curves using Frobenius map, *Public Key Cryptography*, LNCS 1431, S-V, 1999, pp.195-202. [409](#)
5. H.Cohen, *A course in computational number theory*, Graduate Texts in Math. 138, Springer-Verlag, 1993, Third corrected printing, 1996. [408](#)
6. H.Cohen, A.Miyaji and T.Ono, Efficient elliptic curve exponentiation, *Information and Communications Security*, LNCS 1334, Springer-Verlag, 1997, pp.282-290. [408](#)
7. H.Cohen, A.Miyaji and T.Ono, Efficient elliptic curve exponentiation using mixed coordinates, *Advances in Cryptology-ASIACRYPT'98*, LNCS 1514, Springer-Verlag, 1998, pp.50-65. [405](#), [407](#), [408](#)
8. J.Guajardo and C.Paar, Efficient algorithms for elliptic curve cryptosystems, *Advances in Cryptology-CRYPTO'97*, LNCS 1294, Springer-Verlag, 1997, pp.342-356. [405](#)
9. T.Kobayashi, H.Morita, K.Kobayashi and F.Hoshino, Fast elliptic curve algorithm combining Frobenius map and table reference to adapt to higher characteris-

- tic, *Advances in Cryptology-EUROCRYPT'99*, LNCS 1592, Springer-Verlag, 1999, pp.176-189. 405, 409, 412, 416
10. N.Koblitz, Elliptic curve cryptosystems, *Math. Comp.*, 48, 1987, pp.203-209. 405
 11. N.Koblitz, CM curves with good cryptographic properties, *Advances in Cryptology-CRYPTO'91*, LNCS 576, Springer-Verlag, 1992, pp.279-287. 409
 12. K.Koyama and Y.Tsuruoka, Speeding up elliptic cryptosystems using a signed binary method, *Advances in Cryptology-CRYPTO'92*, LNCS 740, Springer-Verlag, 1993, pp.345-357. 405, 408
 13. D.E. Knuth, *The art of Computer Programming: Seminumerical Algorithms*, 3rd edition, Addison Wesley, 1998. 411, 414
 14. C.H.Lim and P.J.Lee, A key recovery attack on discrete log-based schemes using a prime order subgroup, *Advances in Cryptology-CRYPTO'97*, LNCS 1294, Springer-Verlag, 1997, pp.249-263. 409
 15. C.H.Lim and H.S.Hwang, Fast elliptic scalar multiplication with precomputation, preprint, 1999. 407, 409, 418
 16. J.Lopez and R.Dahab, Improved algorithms for elliptic curve arithmetic in $GF(2^n)$, *Selected Areas in Cryptography*, LNCS 1556, Springer-Verlag, 1999, pp.201-212.
 17. J.Lopez and R.Dahab, Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation, *Cryptographic Hardware and Embedded Systems*, LNCS 1717, Springer-Verlag, 1999. 408, 409
 18. W.Meier and O.Staffelbach, Efficient multiplication on certain non-supersingular elliptic curves, *Advances in Cryptology-CRYPTO'92*, LNCS 740, Springer-Verlag, 1993, pp.333-344. 409
 19. V.S.Miller, Use of elliptic curves in cryptography, *Advances in Cryptology-CRYPTO'85*, LNCS 218, Springer-Verlag, 1986, pp.417-426. 405
 20. P.L.Montgomery, Speeding the Pollard and elliptic curve methods of factorization, *Math. of Computation*, 48, 177, 1987, pp.243-264. 408
 21. V.Muller, Fast multiplication on elliptic curves over small fields of characteristic two, *J. of Cryptology*, vol.11, no.4, 1998, pp.219-234. 409
 22. A.Schroeppel, H.Orman, S.O'Malley and O.Spatschek, Fast key exchange with elliptic curve systems, *Advances in Cryptology-CRYPTO'95*, LNCS 963, Springer-Verlag, 1995, pp.43-56. 405, 412
 23. J.A.Solinas, An improved algorithm for arithmetic on a family of elliptic curves, *Advances in Cryptology-CRYPTO'97*, LNCS 1294, S-V, 1997, pp.357-371. 405, 408, 409
 24. E.De Win, A.Bosselaers and S.Vandenberghe, A fast software implementation for arithmetic operations in $GF(2^n)$, *Advances in Cryptology-ASIACRYPT96*, LNCS 1163, Springer-Verlag, 1996, pp.65-76. 405
 25. M.J.Wiener and R.J.Zuccherato, Faster attacks on elliptic curve cryptosystems, *Selected Areas in Cryptography*, LNCS 1556, Springer-Verlag, 1999, pp.190-200. 409, 419
 26. IEEE P1363: Standard Specifications for Public Key Cryptography, *Working Draft*, Oct. 1998. 405, 406, 407, 409
 27. ANSI X9.62: The elliptic curve digital signature algorithm, *Working Draft*, Oct. 1998. 405, 409
 28. ANSI X9.63: Elliptic curve key agreement and key transport protocols, *Working Draft*, Oct. 1998. 405, 409

An Auction Protocol Which Hides Bids of Losers

Kazue Sako

C&C Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae Kawasaki 216-8555 Japan
sako@ccm.cl.nec.co.jp

Abstract. Many auction protocols using practical cryptographic means have successfully achieved capability of hiding the bids of each entity, but not the values of bids themselves. In this paper we describe an auction protocol which hides the bids of non-winners even from the bid-opening centers, and still makes it possible to publicly verify the validity of the winning bid, i.e. that it was the highest bid submitted. The first approach to such a protocol was made by Kikuchi et al in [KHT98]. However, several deficiencies have been pointed out regarding their protocol; for example, it is not well suited for handling tie bids.

We present an auction protocol in which a bid will not be successfully decrypted unless it is the highest bid, thus ensuring bid privacy. In addition, it enables participants to verify that the winning bid is indeed the highest. Also in contrast to the previous work, our protocol can identify all the winners who submitted the winning bid.

Our protocol allows for very compact representations for bids: a bid is represented by a single probabilistic encryption. In the protocol of [KHT98] a bid is represented by a vector of encryptions, of length linear in the number of possible bid values.

We present two practical schemes based on the ElGamal cryptosystem and the RSA cryptosystems, respectively.

Key words: auction, privacy, group decryption.

1 Introduction

Fairness, privacy and correctness have been considered to be three major security issues in auction protocols. By fairness, we mean that we want to ensure that neither the value of a bid itself nor any partial information, e.g. information that might give any bidder an unfair advantage, will be disclosed before the opening time. By privacy, we mean that we do not want to reveal which entity has bid at what value even after the opening. By correctness, we mean that we want the winning bid to be the highest (or lowest) among bids which were entered before opening time, and we want the winner to be the person who made that bid.

These goals have been successfully achieved as depicted in the work of Franklin and Reiter[FR96]. However, no practical protocol has succeeded in keeping the *secrecy of losing bids*, i.e. no protocol that makes winning bid public

but not the losing ones has yet been satisfactorily developed. This is important, as disclosure of these values also reveals information on strategies of losing entities. Leakage of such strategies will strongly influence succeeding auctions with a similar group of entities. However, despite the needs for such a protocol, almost all practical protocols intentionally reveal all the actual values that were bid, in order to enable verification that the winning bid was indeed the highest or lowest among all those made.

In this paper, we describe a protocol that enhances privacy in auctions by keeping values that were bid *secret*, but still enables the fact that the winning value is indeed the highest of the submitted bids to be *publicly* verified.¹

Kikuchi et al [KHT98] first addressed this problem. They try to achieve this property by having a bid represented in a vector of L values, where L is the number of possible values that can be bid. Bidders place 0 on the values they do not wish to bid. The winning bid is determined by adding all the submitted bid-vectors and finding the last non-zero element in the summed vector. The protocol achieves minimum round complexity for bidders, as once they have submitted their bids they do not need to participate in opening. Unfortunately, their scheme, together with an enhanced scheme in [HTK98], is not well suited for handling tie bids, i.e. cases when there are two or more entities who submit the same winning bid. If this happens, their scheme can not specify who the winners are, or even how many winners there are.

Concurrent to our work, Sakurai and Miyazaki proposed a publicly verifiable auction scheme [SM99]. Using the techniques of the convertible undeniable signature scheme[MS97], their scheme succeeds in hiding the losing bids without assuming any trusted centers. The drawback of their scheme is that all bidders must participate in the opening of the bids, by executing a disavowal protocol for each values they did not bid until one finds the highest bid. Thus both computational and round complexity of the bidders are high. Recently, in [KM99], Kobayashi and Morita proposed several auction schemes based on the techniques of the hash chains. It dramatically improves the computational complexity. However, their schemes either require high round complexity to bidders, or require the center to *know* the all bids in order to find the highest one, i.e., the scheme can not keep any bids secret from the center.

We take a totally different approach from all these schemes to achieve bid secrecy and verifiability. Compared to the scheme proposed in [KHT98]², we allow an efficient representation for bids at the cost of computational complexity at the authorities: We require a bidder to post a bid which is an encryption of *one* message, and the authorities to work on it multiple times, whereas the [KHT98] scheme requires a bidder to post multiple messages and the authorities to work on it once. In contrast to the scheme in [SM99], we employ a multiple of centers, who we trust to perform a threshold decryption. This setting helps to keep the round and computational complexity low for the bidders. Unlike the

¹ With a slight modification, the protocol can be made to open the lowest bid.

² We note that Kikuchi et al also aimed at achieving bidder privacy, which is to allow anonymous participation.

one-move auction scheme in [KM99], our scheme can keep the secrecy of the losing bids even from the centers.

Our approach involves a novel usage of encryption and a set of keys. We express a bid as an encryption of a *known* message, with the *key* to encrypt it corresponding to the value bid. Thus, what we hide in ciphertext is not the message that is encrypted, but the key used to encrypt it. The bid itself can be identified by finding the corresponding decrypting key that successfully decrypts to a given message.

The protocol proceeds as follows: each bidder signs and posts an encryption of his bid. At the opening stage, authorities try to find the largest value, the decrypting key of which successfully decrypts one of the submitted bids. Authorities release information on decrypting keys for the winning bid and for the higher values. Then anyone can successfully identify all the winners with the winning bid, and ensure that no one has bid any value higher than the winning bid. Furthermore, the algorithm prevents even the authorities from learning the losing bids.

We present two practical schemes, one based on the ElGamal cryptosystem and one based on the RSA cryptosystem.

2 Previous Work

2.1 The Protocol of [KHT98]

In this section, we describe an abstracted scheme which incorporates the ideas given in [KHT98]. For simplicity, we assume the winners to be the one who has bid the highest value among a set of L possible bid values, $V = \{v_1, \dots, v_L\}$.

We describe how a bidder i with his identity information ID_i would bid a value $v_{b_i} \in V$. The encoding of the value v_{b_i} is represented as a vector of L components, where the first b_i components are independently encrypted ID_i 's and the rest 0. We will call this a bid-vector A_i of a bidder i .

$$A_i[j] = \begin{cases} f_j(ID_i) & \text{if } b_i < j, \\ 0 & \text{otherwise} \end{cases}$$

(Here, f_j is an encryption function for j -th component.) The idea of finding the highest bid is as follows. Given bid-vectors of all the bidders, each elements in the same component are added to generate what we will call a sum-vector T .

$$T[j] = \sum A_i[j]$$

If the last component $T[L]$ is 0, it means no one bid the value v_L . If we search for $j = L, L-1, \dots, 1$ and find the first non-zero value $T[j]$ at $j=t$, then the winning bid is v_t and the winner w is identified by performing $f_t^{-1}(T[t]) = ID_w$. This sum-vector is published for verification.

In order to keep bids secret from authorities, each element $A_i[j]$ in a bid-vector is distributed among authorities using secret sharing techniques[Sha79], which addition of shares yield addition of secrets.

Further, in order to prevent faulty bidder who tries to bid using other entity's ID, the authors of the paper suggest to use ID's which are secretly signed by the authorities.

2.2 Weaknesses in [KHT98]

The protocol described above has the following weaknesses:

- **Protocol failure in a tie case.**

If there were two or more entities who submit the same highest bid t , the value $T[t]$ is an addition of multiple IDs that have been encrypted by f_t . There is no way to decompose this sum to recover original IDs, and thus the protocol fails in identifying the winners.

- **Leaking the second highest bid to a winner.**

A winner would be able to detect the second highest bid, by scanning beyond the t -th component of the sum-vector to find the very next component which is not the encryption of his ID. This contradicts to the aim of revealing no information on the bids of losers.

- **Inefficient bid representation.**

Each bid is represented in a vector of L elements, where L is a number of possible values that can be bid. Further, this long bid-vector is distributed among authorities, so that the representation in shares are proportional to number of authorities.

- **Anonymous interference.**

Anyone can anonymously disturb an auction by submitting a random number r which does not decrypt to his ID.

In the following section, we present a protocol robust against any disturbance from malicious bidders. It successfully identifies all the winners and hides other losing bids, with the same minimum round complexity. Although the computation cost increases at the authorities, the protocol allows a bid to be represented in one probabilistic encryption.

3 The Basic Protocol

3.1 Outline

The basic idea behind our proposed protocol is to present a probabilistic encryption of bid v in such a way that it will not be decrypted unless v is the winning bid. For simplicity, we assume winners to be those who have bid the highest value among a set of L possible bid values, $V = \{v_1, \dots, v_L\}$. We assume multiple authorities open the bids. If it is not necessary to keep bids secret from the authorities, then a setting with a single authority suffices. Note that this case remains nontrivial, since the authority must still prove that the highest bid is indeed the highest.

In order to achieve our goal, we employ a set of encryption functions $\{E_v\}$ and a set of decryption functions $\{D_v\}$ for $v \in V$. The ciphertext of a bid v will

be an encryption $E_v(M_v)$ for a predetermined value M_v . All encrypted bids from each bidder are signed and posted, and authorities will perform decryption to open *only* the highest bid. The opening procedure is as follows: the authorities first take the largest $v_L \in V$ and try to decode all the encrypted bids one by one using D_{v_L} . If any of these decodes to a predetermined value M_{v_L} , it is an indication that the ciphertext was encrypted using E_{v_L} , and thus the bidder is a winner with a winning bid of v_L . If not, then the authorities take the next largest value v_{L-1} and continue until they find the largest v_t for which at least one of the encrypted bids indeed decodes to M_{v_t} .

3.2 Sets of Encryption and Decryption Functions

We require the following properties on the function sets $\{E_v\}$ and $\{D_v\}$ and a set of values $\{M_v\}$:

Property 1 *Indistinguishability*

Given any $E_v(M_v)$ and $E_{v'}(M_{v'})$ for $v, v' \in V$, a polynomial turing machine can not distinguish whether or not $v = v'$.

Property 2 *Incompatible decryption*

Given any $E_v(M_v)$ and $v' \in V$ that is equal to or larger than v , $D_{v'}(E_v(M_v)) = M_{v'}$ if and only if $v = v'$.

Property 3 *Independent decryption*

Given any $E_v(M_v)$ and $v' \in V$ that is strictly larger than v , $D_{v'}(E_v(M_v))$ does not give any information on v , except that $v \neq v'$.

Property 4 *Group decryption*

Each decryption function D_v is distributed among authorities, such that decryption is possible only through a collaboration of authorities forming a quorum. (This property is not necessary if keeping bids secret from the authorities is not required.)

Property 5 *Verifiable decryption*

Given $E(M)$ and M' and E_v , the authorities can supply a proof that proves that M' is a correct decryption of $E(M)$ under decryption function D_v .

Property 6 *Verifiable generation*

We require a means to verify that the sets $\{E_v\}$, $\{D_v\}$ and $\{M_v\}$ have been chosen to achieve the above properties.

3.3 Bidding and Opening

Given the set of functions satisfying the above properties, the auction proceeds as follows:

1. [Set-up] The authorities set up $\{E_v\}$, $\{D_v\}$ and $\{M_v\}$, where $\{E_v\}$ and $\{M_v\}$ are posted in a way that anyone can confirm their validity(Property 6).
2. [Bidding] Each bidder b with a bid v_b posts $C_b = E_{v_b}(M_{v_b})$ with his signature.
3. [Opening] The authorities

- (a) set $k = L$ and decrypt $\{C_b\}_b$ using D_{v_k} .
 - (b) While $D_{v_k}(C_b) \neq M_{v_k}$ for all b , set $k = k - 1$.
 - (c) Publish $t = k$ as the winning bid and list all b s.t. $D_{v_t}(C_b) = M_{v_t}$ as winners $\{w_i\}$.
 - (d) Publish for all $j \geq t$, $D_{v_j}(C_b)$ with its proof of being correct(Property 5).
4. [Verification] Anyone may verify the following:
- (a) $D_{v_j}(C_b)$ is the correct decryption regarding each D_{v_j} , for $j \geq t$.
 - (b) For all $v_j \neq v_t$, $D_{v_j}(C_b) \neq M_{v_j}$.
 - (c) For each of the winners w_i , $D_{v_t}(C_{w_i}) = M_{v_t}$ holds.

3.4 Discussions

We claim that the following properties are achieved in this protocol.

– *Correctness*

The winning bid is indeed the highest among all the bids. If there exists a k that is larger than the winning bid t , then the authorities should have stopped when scanning the bids with D_k (Property 2). Winners are those who submit the winning bid, as correctness of the opening is also guaranteed in Property 2.

– *Verifiability of the result*

Due to the Property 5, all entities can verify that the authorities performed the correct decryptions for $k \geq t$. Therefore, everyone can confirm that there is no bid higher than t and that the announced winners are the only ones who submitted the winning bid t .

– *Fairness*

In order to achieve fairness, we require our encryption to be non-malleable [DDN91]. Informally, this property ensures that seeing one bidder's encryption does not give another bidder an unfair advantage, say by generating an encryption of a bid that is one dollar more than the previous bid. We note that we can add nonmalleability by reencryption: If an encryption function E_v does not by itself achieve non-malleability, we can always re-encrypt using some arbitrary non-malleable encryption. This encryption can be removed prior to decrypting via E_v .

We note that a nonmalleable encryption is still vulnerable to a replay attack, posting the same bid. Some fixes to this problems are: 1. Do not accept the second same bid, or 2. Encrypt $M_{vb}||b$, where b is the identity of the bidder, or 3. Provide proof of ownership of the ciphertext that the bidder himself has generated it.

– *Non-repudiation*

The winners can not deny they submitted the winning bid, as there is a digital signature given to their encrypted bid which indeed decrypts properly.

– *Privacy of losing bid*

Due to Property 1, the value of bids is hidden in the encryption. Due to Property 4, the bids of losers will not be revealed in the course of opening,

that is, when the decryption functions are performed on them. Once the winning bid is found, then no further decryption trial will be performed on the losing bids, so they are never decrypted.

- *Robustness*

Even if an invalid bidder submits a meaningless encryption, the auction proceedings will be unaffected; the invalid bids are simply ignored.

- *Efficiency*

Bidders need to encrypt only once and submit only a single encryption. On the other hand, authorities need to perform as many as $(L - t + 1)B$ decryptions, where t denotes the winning bid v_t and B is the number of bidders. The authorities further need to publish proofs for decryptions, the cost of which varies depending on the implementation.³

4 Schemes Based on Practical Cryptosystems

In the following, we give two examples of the set of functions achieving the property discussed in Sect. 3.2. They are based on the ElGamal cryptosystem and the RSA cryptosystem, respectively.

ElGamal based ones require a list of public keys $\{E_v\}$ for each $v \in V$. On the other hand, RSA based ones allow bidders to generate $\{E_v\}$ from the value v , so this long list is not necessary. However, in the aspect of information necessary verify the decryption, the ElGamal based schemes require authorities to reveal only the corresponding secret keys whereas the RSA based schemes require them to publish each decryption results. Further, the ElGamal based schemes are suited to prove that they achieve required properties such as indistinguishability and incompatible decryption property, whereas we can only heuristically claim such in RSA based schemes. Procedures to distributedly generate keys among authorities are more complicated in the RSA scheme than the ElGamal scheme.

4.1 ElGamal Based Scheme

In this section, we use a set of encryptions based on the ElGamal cryptosystem [E84]. We assume a large prime p where $p - 1$ has a large prime factor q is given by the authorities together with a generator $g \in Z_p^*$ over a subgroup of order q . Further, z_v is independently and disjointly generated for each $v \in V$, which will be secretly held by the authorities. For simplicity let $M_v = M$ for all v . We define E_v by

$$E_v(M) = (g^\alpha \bmod p, M \cdot h_v^\alpha \bmod p)$$

where $h_v = g^{z_v} \bmod p$ is a public parameter for value v and α is a non-zero random number in Z_q .

³ For example, the ElGamal-based scheme in Subsect. 4.1 requires $L - t$ keys to be published, whereas the RSA-based one in Subsect. 4.2 requires $(L - t + 1)B$ decryption results.

Given $E(M) = (x, y)$, we define D_v by

$$D_v(E(M)) = y/x^{z_v} \bmod p$$

We will show that the above sets of functions fulfill our requirements.⁴

- [Property 1] *Indistinguishability*

Given any $E_v(M) = (x, y)$ and $E_{v'}(M) = (x', y')$ for $v, v' \in V$, finding $v = v'$ or not is equivalent to determining whether quadruples $(x, x', y/M, y'/M)$ are random or $\log_x(y/M) = \log_{x'}(y'/M)$ holds. This is as difficult as the Diffie-Hellman Decision Problem(c.f. [CS98]).

- [Property 2] *Incompatible decryption*

Given any $E_v(M)$ and $v' \in V$, $D_{v'}(E_v(M_v)) = M \cdot g^{\alpha \cdot (z_v - z_{v'})}$. This equals M only if or $z_v = z_{v'}$, since $\alpha \neq 0 \bmod q$. Since the z_v 's are disjointly generated, it follows that $v = v'$.

- [Property 3] *Independent decryption*

Given any $E_v(M)$ and $v' \in V$, $D_{v'}(E_v(M_v)) = g^{\alpha \cdot (z_v - z_{v'})}$ indeed does not reveal any information on v , except that $v \neq v'$, as long as z_v 's are generated randomly.

- [Property 4] *Group decryption*

The key pair (z, h) is constructed in a way that each authority receives a share z_i and is publicly committed to this share by $h_i = g^{z_i}$ [Ped91].

- [Property 5] *Verifiable decryption*

Given $E(M) = (x, y)$ and h_v , authorities can publish the secret key z_v . Then one can perform decryption by themselves for any $E(M)$ by $M = y/(x^{z_v})$.

- [Property 6] *Verifiable generation*

By an appropriate use of pseudo-random generators, we can confirm that the set $\{E_v\}$, $\{D_v\}$ and $\{M_v\}$ are chosen independently, and thus suffices the above properties.

4.2 RSA Based Schemes

In this subsection, we use a set of encryptions based on the RSA cryptosystem [RSA]. In contrast to the protocol described in the previous subsection where we need to publish a list of public parameters for all possible bid v , the protocol below allows bidders to generate public parameters for themselves.

We require the authorities to generate two large primes p and q where $p - 1$ and $q - 1$ can be represented as $2^k p'$ and $2^\ell q'$ respectively for large prime factors p' and q' . The product $N = p \cdot q$ is published, together with a cryptographically secure hash function H ⁵.

⁴ We note it does not achieve non-malleable property as it is. A simple fix is to apply an encryption that is known to achieve non-malleability, on top of the encrypted bids. Another heuristic approach is to include bidder's ID in the message, e.g., $M|ID|\text{hash}(M|ID)$.

⁵ For the defined encryption scheme to be secure against adaptive chosen message attack, we require the hash function H to be division intractable [GHR99].

We define M_v to be

$$M_v = v|R|H(v|R)$$

where R is a random number of predetermined bit length and $|$ is a concatenation.⁶
The encryption E_v is given by

$$E_v(M_v) = (v|R|H(v|R))^{H(v)|1} \bmod N.$$

The decryption algorithm D_v proceeds as follows:

1. First, compute an odd exponent $e_v = H(v)|1$. This exponent is almost certainly relatively prime to $LCM(p-1, q-1)$.
2. Compute $d_v = e_v^{-1} \bmod LCM(p-1, q-1)$.
3. Decrypt $E(M)$ by computing $\{E(M)\}^{d_v} \bmod N$. One can conclude $E(M)$ to be an encryption for the bid v if the decrypted output is conformant with $v|R|H(v|R)$.

We will informally argue that the above sets of functions and parameters should fulfill our requirements.

- [Property 1] *Indistinguishability*

Given any $E_v(M_v)$ and $E_{v'}(M_{v'})$ for $v, v' \in V$, we know of no efficient way of determining whether or not $v = v'$ holds.

- [Property 2] *Incompatible decryption*

Given any $E_v(M_v)$ and $v' \in V$, $D_{v'}(E_v(M_v))$ is not likely to yield a decrypted message M with prefix v' unless $v = v'$.

- [Property 3] *Independent decryption*

Given any $E_v(M_v)$ and $v' \in V$, $D_{v'}(E_v(M_v))$ would not provide a useful information to reveal v , except that $v \neq v'$.

- [Property 4] *Group decryption*

The methods to generate and share RSA keys in a distributed manner are studied in [FMY98, MS99].

- [Property 5] *Verifiable decryption*

For each $E(M)$ and E_v , the authorities can reveal $M_v = D_v(E(M))$. It is easy to verify that M_v is indeed a decryption of $E(M)$ under D_v by simply checking if $(M_v)^{H(v)|1} = E(M)$, and M_v conforms to $v|R|H(v|R)$. Note that on contrast to the ElGamal based schemes, we can not give away d_v which is a decryption exponent. A pair d_v and e_v gives prime factors of N which makes all encryption decrypted.

- [Property 6] *Verifiable generation*

The sets $\{E_v\}$, $\{D_v\}$ and $\{M_v\}$ suffice above properties, if we can verify that the RSA keys are properly generated in a distributed manner.

⁶ R can contain the bidder's ID for non-malleability purpose.

5 Conclusion

In this paper we proposed an auction protocol which hides the bids of non-winners, and still makes it possible to publicly verify that the winning bid was indeed the highest submitted. Our protocol enjoys minimum round complexity for bidders and does not suffer from any problems in case of a tie bid, which was the case in previous work. Moreover, our protocol is efficient for bidders in that it requires only a single encryption for each bid, whereas in previous work the length of encoded bid was required to be linear to the number of possible bid values. The uniqueness of our construction is that we hide the index of keys used in encryption rather than the message.

We gave two concrete examples of schemes, one based on the ElGamal cryptosystem, and the other on the RSA cryptosystem.

Acknowledgements

The author would like to thank Joe Kilian for his valuable comments to this work, including the suggestion of the RSA-variant of the proposed scheme. She also thanks Hiroaki Kikuchi and Shingo Miyazaki for the discussions on this topic. Her thanks also goes to the anonymous referees for the invaluable comments on the paper.

References

- [CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, 1998. [429](#)
- [DDN91] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *STOC '91*. [427](#)
- [E84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, 1984. [428](#)
- [FMY98] Y. Frankel, P. MacKenzie and M. Yung, *Robust efficient distributed RSA-key generation*. in *STOC 98*, pp.663-672,1998. [430](#)
- [FR96] Matthew Franklin and Michael Reiter. *The design and implementation of a secure auction service*. In *IEEE Transactions on Software Engineering*, No.22, Vol.5, pages 302-312, 1996. [422](#)
- [GHR99] Rosario Gennaro, Shai Halevi and Tal Rabin. *Secure hash-and-sign signatures without the random oracle*. In *Eurocrypt 99*, 1999. [429](#)
- [HTK98] Michael Harkavy, Doug Tyger, Hiroaki Kikuchi. *Electronic auctions with private bids*. In *Third USENIX Workshop on Electronic Commerce*, 1998. [423](#)
- [KHT98] Hiroaki Kikuchi, Michael Harkavy, Doug Tyger. *Multi-round anonymous auction protocols*. In *IEEE Workshop on Dependable and Real-Time E-Commerce System* 1998. [422](#), [423](#), [424](#), [425](#)

- [KM99] Kunio Kobayashi and Hikaru Morita. *Efficient sealed-bid auction with quantitative competition using one-way functions*. In *Technical Report of IEICE, ISEC* May 1999. 423, 424
- [MS97] M. Michels and M. Stadler. *Efficient convertible undeniable signature schemes*. In *Proc. 4th Annual Workshop on Selected Areas in Cryptograph, SAC'97*, 1997. 423
- [MS99] Shingo Miyazaki and Kouichi Sakurai. *Notes on threshold schemes in the distributed RSA Cryptosystem*. In *the 1999 Symposium on Cryptography and Information Security*, pp.451–456, 1998. 430
- [Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, 1991. 429
- [RSA] Ronald Rivest, Adi Shamir and Len Adleman. *A method for obtaining digital signature and public-key cryptosystems*. In *Communications of the ACM*, Vol.21, No.2, pages 120–126. 1978. 429
- [SM99] Kouichi Sakurai and Shingo Miyazaki. *A bulletin-board based digital auction scheme with bidding down strategy –towards anonymous electronic bidding without anonymous channels nor trusted centers* in *Cryptographic Techniques and E-Commerce, Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce(CryTEC '99)* 1999. 423
- [Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979. 424

Forward Secrecy and Its Application to Future Mobile Communications Security*

DongGook Park^{1,2}, Colin Boyd² and Sang-Jae Moon^{3**}

¹Korea Telecom, Access Network Laboratory,
17 WooMyeon-Dong, SeoCho-Gu, 137-792, Seoul, Korea
park@isrc.qut.edu.au

²Queensland University of Technology, Information Security Research Centre,
2 George Street, GPO Box 2434, Brisbane, Queensland 4001, Australia
boyd@fit.qut.edu.au

³Kyungpook National University, School of Electronic & Electrical Engineering,
1370, SanKyuk-Dong, Pook-Gu, Taegu, 702-701, Korea
sjmoon@ee.knu.ac.kr

Abstract. Perfect forward secrecy, one of the possible security features provided by key establishment protocols, concerns dependency of a session key upon long-term secret keys (symmetric or asymmetric). The feature promises that even if a long-term private key is disclosed to any adversary, the session keys established in the protocol runs using the long-term key would not be compromised. The importance of this kind of belief may differ greatly among application environments, in terms of both communication types and different communicating entities. We describe two generic prototypes of protocols which bring forward secrecy to security protocols. We note that future generation mobile communication environment will be filled with diverse types of communication users and data. The security protocol in a prominent future mobile system, UMTS, was originally designed without any consideration of perfect forward secrecy. We consider modified protocols to provide this property.

1 Introduction

The use of session keys allows different sessions to be independently secure so that if one session key becomes compromised then this should not affect any other session key. Long term keys (symmetric or asymmetric) are used to establish session keys and so must be protected much more securely than session keys. Although it may be

* This research is part of the co-operative project “Security Technologies in Wireless Communications” between Queensland University of Technology and Korea Telecom

** This work was done while this author was a visiting research fellow at the Information Security Research Centre, Queensland University of Technology, Brisbane, Australia.

an unlikely event, the consequences of the compromise of a long-term key should be considered.

A protocol is said to provide *forward secrecy* if the compromise of long-term keys does not compromise past session keys that have been established before the compromise of the long-term key [3]. The idea of forward secrecy seems to have been coined by Günther [4] in connection with an identity based protocol he proposed. In fact Günther used the term *perfect* forward secrecy; however since the word ‘perfect’ has connotations with unconditional security which are not relevant here, we will use the simpler term in common with a number of other authors. It should be noted that there seems to be a disagreement in the definition of forward secrecy because we can find a literature where forward secrecy is intended to mean that a secret encryption key used in a session must be securely discarded after the session to prevent an adversary from obtaining the encryption key in any way and eavesdropping any future sessions protected by the same encryption key [7]. In this paper, however, we use only the former definition of forward secrecy, which appears more generally agreed one.

We also note that there is a somewhat similar concept called forward *security* to address another significance of losing long-term private keys [9]. A signature scheme with forward security protects users from the threat of *signature forgery* in case their signature keys have been compromised. The basic idea to implement forward security is to update the signature key itself frequently to reduce the risk of key exposure. This may contributes also to the forward *secrecy* when it is the case that the signature key is used for authentication and key establishment as well, because the limited longevity of the signature key reduces the risk of relevant session key compromise down to the lifetime of the signature key. Still, however, forward security is not a sufficient condition for forward secrecy considering that the disclosure of the signature key would compromise any session keys computed using the signature key. In other words, if we confine our focus on a particular long-term private key (however long it lives), then it is only forward secrecy that protects the relevant session keys from the compromise of the long-term private key. With this in mind, we argue that the essential characteristic of forward secrecy is orthogonal to that of forward security. It should be noticed, however, that there seems to be rather loose distinction, which reflects, as we have already described, the fact that forward security may be regarded as a weak alternative to forward secrecy in a practical sense [2]. In this paper, however, we confine our discussion only to the forward secrecy in distinction to forward security.

It has long been known that protocols based on the Diffie-Hellman key agreement protocol [12] will usually provide forward secrecy. This is because the long-term keys are normally used only to authenticate messages and not to encrypt them. This property is widely regarded as a useful extra security feature of Diffie-Hellman based protocols, since most other protocols do not possess it. In the next section, however, we will see that forward secrecy does not require any particular type of cryptosystem such as Diffie-Hellman. Considering the significance of forward secrecy, it is rather surprising that this security feature has almost never been given a proper effort to understand it.

Unlike many other goals of security protocols, forward secrecy may have to be treated more practically. Its significance in the real applications dramatically varies

through both angles of communication types and user types. In the communications between a private user and a public commercial entity, it is more the user than the commercial entity that is concerned about confidentiality for the past communications, and hence is more concerned about forward secrecy. On the other hand, forward secrecy usually requires some additional computations of asymmetric key cryptography, and hence might be a quite expensive cryptographic service in some types of communications, for example, voice communications or message broadcasting in some value added services.

In the next section two prototype constructions are presented for protocols providing forward secrecy. The first is based on the Diffie-Hellman protocol, while the second can be used with any chosen asymmetric encryption scheme. Section 3 then discusses the notion of partial forward secrecy and presents prototype constructions. In sections 4 and 5 we examine a prominent proposed protocol for third generation mobile communications, show that it does not provide forward secrecy, and show how it may be modified to do so.

Now, we summarize below the notation used in this paper.

A, B:	the identities of the two peer principals involved in a particular session
g:	a generator of a finite group
r_A, r_B:	random nonces chosen by the principals, A and B , respectively
K_{AB}:	a secret session key established between two principals, A and B
b:	the private key component of the public-private key-agreement key pair of the principal, B
g^b:	the public key component of the public-private key-agreement key pair of the principal, B
{m} _{K_A} ⁻¹ :	the message m signed by the user with his/her private signature key K _A ⁻¹
{m} _{K_{AB}} :	the symmetric encryption of a message m using the session key K _{AB}
h:	a common hash function agreed between the two principals, A and B
e_A, n_A:	a temporary asymmetric public key pair of A in the RSA context

To make some description clear, we also use novel but easy-to-understand notation, which we want to help capture some abstract property with regard to forward secrecy, as follows.

APriKey, APubKey:	a <i>long-term certified</i> authentic asymmetric key pair (private key, APriKey and public key, APubKey) of a principal A
APriKeyX, APubKeyX:	a <i>temporary uncertified</i> asymmetric key pair (private key, APriKeyX and public key, APubKeyX) of a principal A
APubKey{m}:	a message m encrypted under the <i>long-term authentic</i> public key, APubKey of A for data confidentiality. The message m can only be retrieved by the owner of the APriKey (i.e., the principal A)
APubKeyX{m}:	a message m encrypted under the <i>temporary</i> public key of A for data confidentiality. The message m can only be retrieved by the owner of the APriKeyX (i.e., the principal A)

2 Two Prototypes for Forward Secrecy

Almost all authors discuss only Diffie-Hellman based key establishment protocols as examples providing forward secrecy and there seems to be an implicit assumption that these are the only possible examples. Consequently, there seems to be a tendency in protocol design that only Diffie-Hellman type key agreement functions are considered for forward secrecy implementation [1]. In this section, two prototype constructions are presented, one based on the special algebraic properties of Diffie-Hellman key exchange, and the other which can work with any asymmetric encryption scheme. The second prototype addresses the question: *is there any other cryptographic algorithm than Diffie-Hellman satisfying forward secrecy?* We do not try to answer this question directly. Instead, we enlarge our scope of the search for forward secrecy to cover cryptographic *protocols*. Then, the answer to the above question is: “Yes. As many as the number of different asymmetric cryptographic algorithms.”

We take a rather over-simplified approach to ease the capture of the very basic property which enables forward secrecy. Any other security goals such as entity authentication and key authenticity will be entirely omitted in the following descriptions. We believe that by taking this approach we can understand the mechanism of forward secrecy more clearly. Furthermore, these building blocks of forward secrecy mechanisms without any extra property would be more effectively integrated into authentication and key establishment protocols.

We first investigate what kind of property in Diffie-Hellman type protocols present us forward secrecy. Two principals **A** and **B** select random secrets r_A and r_B , compute g^{r_A} and g^{r_B} respectively, and exchange them over an unsecured channel.

1. $A \rightarrow B: g^{r_A}$
2. $A \leftarrow B: g^{r_B}$
$A: K_{AB} = (g^{r_B})^{r_A}$
$B: K_{AB} = (g^{r_A})^{r_B}$

Fig. 1. Basic Diffie-Hellman key agreement

This basic protocol for key agreement is integrated into more sophisticated protocols which use long-term asymmetric keys of principals to provide entity authentication. The famous STS (Station-to-Station) protocol [13] is such an example (Fig. 2).

In the STS protocol, the session key establishment is exactly in the same form as that of basic Diffie-Hellman protocol, and does not depend on the long term asymmetric keys of **A** and **B**. Therefore a future possible disclosure of the long-term keys of **A**, **B** or both does not lead to the compromise of the session key. It should be noted that there are a number of other methods to incorporate authentication into basic Diffie-Hellman and which also provide forward secrecy. Some examples may be found in the IEEE P1363 draft standard [8].

1. $\mathbf{A} \rightarrow \mathbf{B}: g^{r_A}$
 2. $\mathbf{A} \leftarrow \mathbf{B}: g^{r_B}, \{ \{ g^{r_B}, g^{r_A} \}_{K_B^{-1}} \}_{K_{AB}}$
 3. $\mathbf{A} \rightarrow \mathbf{B}: \{ \{ g^{r_A}, g^{r_B} \}_{K_A^{-1}}, \mathbf{A} \}_{K_{AB}}$
- $\mathbf{A}, \mathbf{B}: K_{AB} = g^{r_A r_B}$

Fig. 2. Station-to-Station (STS) protocol

Now, we try to describe the very general property of Diffie-Hellman protocol in a quite abstract way. Using the abstract notation shown in Section 1, we can re-describe the Diffie-Hellman protocol as follows, which we call the first prototype for forward secrecy (Fig. 3).

1. $\mathbf{A} \rightarrow \mathbf{B}: \mathbf{APubKeyX}$
 2. $\mathbf{A} \leftarrow \mathbf{B}: \mathbf{BPubKeyX}$
- $\mathbf{A}: K_{AB} = F(\mathbf{BPubKeyX}, \mathbf{APriKeyX})$
 $\mathbf{B}: K_{AB} = F(\mathbf{APubKeyX}, \mathbf{BPriKeyX})$

where F is a common key agreement function with the following property:

$$F(\mathbf{BpubKeyX}, \mathbf{APriKeyX}) = F(\mathbf{APubKeyX}, \mathbf{BPriKeyX})$$

Fig. 3. An abstract level description of basic Diffie-Hellman key agreement

The public components, **APubKeyX** and **BPubKeyX** of the temporary uncertified asymmetric key pairs of both principals are transmitted in clear, and no long-term key is involved in the calculation of the session key. Only the principals **A** and **B** which are in possession of secret private components of the short-term asymmetric key pairs, i.e. **APriKeyX** and **BPriKeyX**, respectively, can derive the true authentic session key K_{AB} . Another rather trivial requirement for forward secrecy is that the short-term secrets of both parties should be securely discarded after the completion of the corresponding session.

Now, we consider another alternative protocol for forward secrecy, which is also described using the same abstract level notation (Fig. 4).

1. $\mathbf{A} \rightarrow \mathbf{B}: \mathbf{APubKeyX}$
 2. $\mathbf{A} \leftarrow \mathbf{B}: \mathbf{APubKeyX}\{r_B\}$
- $\mathbf{A}, \mathbf{B}: K_{AB} = h(\mathbf{APubKeyX}, r_B)$ or alternatively
 $K_{AB} = r_B$ when key transport scheme is preferred.

Fig. 4. An abstract level description of an alternative protocol for forward secrecy

This second prototype for forward secrecy is based on *confidentiality of a random nonce r_B* chosen by the principal **B**. Here, the temporary public key **APubKeyX** is used for temporary encryption of r_B . On receipt of this encrypted r_B , the principal **A**

can decrypt and recover \mathbf{r}_B using **APriKeyX** as the decryption key. This ephemeral characteristic of the encryption of a random secret is the source of forward secrecy.

The critical difference between the previous prototype derived from Diffie-Hellman scheme and this one is that the former depends on the special feature of a key agreement function \mathbf{F} with an elegant symmetric property, whereas the latter relies upon confidentiality using a temporary public key. Therefore, the first prototype can be implemented only by a cryptosystem which satisfies the requirement of the key agreement function \mathbf{F} . Presently, only Diffie-Hellman key exchange in various suitable groups is known to be such a system.

On the other hand, the second prototype can easily be applied to *any asymmetric cryptosystem*, which does not have to be Diffie-Hellman system. The random nonce \mathbf{r}_A for temporary encryption from **A** to **B** may also be used as a challenge value for **A** to authenticate **B**. The response value then has to be an indication that **B** has used its own secret private key to generate it.

It is interesting to note that the second prototype can be deployed into both key agreement and transport schemes, unlike the first one which allows only key agreement.

We consider two instance protocols which are easily derived from the second prototype. The following protocols are an application of the prototype to discrete log based cryptosystem (Fig. 5) and RSA cryptosystem (Fig. 6) respectively.

1. $\mathbf{A} \rightarrow \mathbf{B}: g^{r_A}$
2. $\mathbf{A} \leftarrow \mathbf{B}: g^{r_A r_B}$
$\mathbf{A}: K_{AB} = h(g^{r_A}, g^{r_B}) = h(g^{r_A}, (g^{r_A r_B})^{r_A^{-1}})$
$\mathbf{B}: K_{AB} = h(g^{r_A}, g^{r_B})$

Fig. 5. Discrete log cryptography based implementation of the second prototype

1. $\mathbf{A} \rightarrow \mathbf{B}: e_A, n_A$
2. $\mathbf{A} \leftarrow \mathbf{B}: (r_B)^{e_A} \bmod n_A$
$\mathbf{A}, \mathbf{B}: K_{AB} = h(e_A, r_B)$

Fig. 6. RSA cryptography based implementation of the second prototype

Even though the example protocol shown in Fig. 5 is very similar to the original Diffie-Hellman protocol, it should be noticed that this is just an example of the prototype, and any different kind of public key mechanisms can be adopted to provide forward secrecy. Of course, as with the prototype protocols, these examples must be used together with authentication in order to achieve a *secure* protocol. It is worth noting that the authentication required can be achieved through either symmetric or asymmetric cryptography, so it should not be assumed that forward secrecy requires asymmetric long-term keys. Indeed, several password based protocols [5], [11], in which the long-term key is a (short) shared secret, or a value derived from it, have been proposed and provide forward secrecy. Whilst it may be of questionable value to

consider the efficiency of incomplete protocols, it is also interesting to note that the example based on RSA can be more efficient than that based on discrete log for entity **B** if the value e_A is chosen to be a small value such as is often recommended. However, we must note that a new temporary RSA key pair must be generated by **A** for each session.

In fact, we have noticed that basically the same concept of forward secrecy based upon RSA had already been described by Wiener [10]. His description, however, still only concerns the implementation of forward secrecy in terms of a particular cryptographic *algorithm* like RSA rather than *protocols* using any asymmetric cryptography. This is likely to lead to a rather misleading conclusion that RSA (or even the second prototype itself) is always a more expensive way to achieve forward secrecy than Diffie-Hellman (or the first prototype). Furthermore, Wiener seems to ignore the possibility that the use of ephemeral encryption for forward secrecy (i.e., the second prototype above) can be just as efficient as the first prototype even in the case of discrete-log cryptosystems. As will be made clear later in this paper, however, both prototypes may require the same cost for forward secrecy to be implemented into key establishment protocols. The choice of the suitable prototype for forward secrecy may be made according to the application scenario rather than the particular cryptographic primitives used in the protocols.

It is an obvious question to ask whether there are other protocols providing forward secrecy that do not fit into the two prototype constructions discussed in this section. It seems that forward secrecy can be provided only through use of a *one-way function*, so that later the session key cannot be recovered because this function cannot be inverted. Because of its special algebraic properties, exponentiation modulo a prime is a suitable one-way function. Naturally other similar functions, such as multiplication in an elliptic curve group can equally be used, and these provide natural generalisations of the original Diffie-Hellman key exchange protocol. Therefore any other one-way function with suitable algebraic properties may be candidates for alternative protocols providing forward secrecy, although there are no such functions currently available to our knowledge. The second prototype makes use of the trapdoor one-way function underlying any asymmetric cryptosystem. In this case the function cannot be inverted once the trapdoor is deleted. Therefore, although we offer no further evidence, we conjecture that forward secrecy can only be provided by protocols which either have similar algebraic properties as modular exponentiation (prototype one) or use trapdoor one-way functions (prototype two).

3 Two Prototypes for *Partial Forward Secrecy*

It seems that we do not need to insist only upon *complete* forward secrecy in some application environments. Instead, we may consider a sort of *imperfect* or *partial* forward secrecy if it is more computationally effective. There may be, for example, a particular communication type where only one of two peer entities is really concerned about confidentiality of the past data and/or the other entity's long-term private key is more likely to be compromised. In this situation, it may be reasonable to consider

protection against the long-term key compromise of only one principal of two peers, which we may call *partial* forward secrecy as opposed to the usual *complete* forward secrecy.

The same reasoning for prototypes of forward secrecy can be directly applied to identify the prototypes for partial forward secrecy. The two prototypes shown in Fig. 7 and Fig. 8 are partial forward secrecy analogues of the previous ones.

1. A \leftarrow B: BPubKeyX

A: $K_{AB} = F(BPubKeyX, APriKey)$
B: $K_{AB} = F(APubKey, BPriKeyX)$

where F is a common key agreement function with the following property:

$$F(BPubKeyX, APriKey) = F(APubKey, BPriKeyX)$$

Fig. 7. The first prototype for partial forward secrecy

1. A \leftarrow B: APubKey{r_B}

A, B: $K_{AB} = h(m, r_B)$ or alternatively $K_{AB} = r_B$ when key transport scheme is preferred.

Notation :

m: an optional key input data to the hash function
(m should be agreed previously between A and B in the more broad context of the relevant whole key establishment protocol which includes this forward secrecy scheme)

Fig. 8. The second prototype for partial forward secrecy

Here in both prototypes, we assume that **B** has an authentic copy of **A**'s long-term public key, **APubKey**. The only difference from *complete* versions is that **A**'s short-term private key is replaced with its long-term authentic public key. The computation procedures of the session key, K_{AB} take **A**'s long-term public key, but not that of the principal **B**. Therefore, the compromise of **B**'s long-term private key alone does not cause the compromise of the session key. It should be noticed, however, that these prototypes, including both partial and complete cases, do not guarantee any other security goals except forward secrecy. It is the whole integration of elements in a protocol that provides the ultimate security goals including entity authentication.

4 Future Mobile Communications and Forward Secrecy

Wireless mobile communications are notorious with so-called *usage fraud* and *eavesdropping* because of its radio media characteristics. Although rather limited, user authentication and data encryption is provided over the air interface of the current generation of digital cellular systems such as Global System for Mobile communications

(GSM). Their security technologies came from conventional symmetric cryptography. It is expected that future generation mobile communication systems such as Universal Mobile Telecommunications System (UMTS) will include asymmetric cryptography to enlarge their security features. The ASPeCT project for research and development of security technologies to be used in UMTS, has proposed a public key based authentication and key establishment protocol with both air interface between the user and the network, and the user-to-Value Added Service Provider (VASP) interface in mind [6]. User to VASP communications will be much like the present wireline based internet communication using web browsers. According to the increase in the number and diversity of VASPs, the trust relations between communicating parties will be dramatically complicated and the deployment of the required public key infrastructure would be a great challenge.

A: user, **B:** network or VASP

1. $\mathbf{A} \rightarrow \mathbf{B}: g^{r_A}$
 2. $\mathbf{A} \leftarrow \mathbf{B}: r_B, h2(K_{AB}, r_B, B)$
 3. $\mathbf{A} \rightarrow \mathbf{B}: \{ \{ h3(g^{r_A}, g^b, r_B, B, K_{AB}) \}_{K_A^{-1}}, A \}_{K_{AB}}$
- $$\mathbf{A}, \mathbf{B}: K_{AB} = h1(r_B, g^{br_A})$$

Notation:

$h1, h2, h3$: one-way hash functions specified for AS-PeCT protocol

Fig. 9. ASPeCT protocol for authentication and key establishment in UMTS

The ASPeCT protocol is basically the same as the Station-to-Station protocol in that both protocols use the same challenge-response mechanism, i.e., **A** and **B** challenge each other with random nonces (g^{r_A} and r_B in this protocol) exchanged in clear and calculate responses using private keys (K_A^{-1} and b respectively in this protocol). In this protocol, however, it should be noted that the second message in the protocol does not contain any signature by **B**. The third message includes the signature by **A** like STS protocol to accommodate non-repudiation requirement for user-network or user-VASP applications.

Significant effort was expended to make this protocol satisfy a demanding set of security goals and computation efficiency at the same time. Through a saving of the non-repudiation property of **B** to **A**, the resultant computational load is significantly lower than that of STS protocol. One additional feature omitted from this protocol is forward secrecy which is supported in STS protocol. This difference comes from the session key generation of the two protocols. The ASPeCT protocol uses a similar but subtly different method from the Diffie-Hellman key computation. STS protocol complies with the original Diffie-Hellman form $g^{r_A r_B}$ where two terms of the exponents are the random nonces generated within two principals. Instead, ASPeCT protocol uses only one nonce r_A and the private key b of **B**. In this way, the protocol saves some public key based computations and succeeds in turning on-line exponentiation

within the user terminal **A** into off-line because **A** may take the advantage of the pre-knowledge of **B**'s public key g^b in most cases. For the sake of simplicity, we only refer to [6] for detailed security analysis of the protocol.

If the long term private key b of the VASP (or network) is compromised and all the protocol transcripts for a particular session are recorded (for the knowledge of g^{r_A} and r_B) by an attacker, the session key for the session is easily disclosed to the attacker, and so no forward secrecy is provided in this protocol. Of course, the disclosure of the private key of the mobile side alone does not lead to the disclosure of the session key (so partial forward secrecy is satisfied), but in that case, the real problem is more authentication rather than forward secrecy. Moreover, *forward secrecy concerns the user who cannot ensure that the private key of the VASP would not be compromised.*

It is surprising that the ASPeCT analysis of protocols has not considered the issue of forward secrecy at all. The likelihood of compromise of a long-term key is difficult to assess. With regard to a trusted network base station this probability may be regarded as sufficiently low. However, when it concerns any VASP, users may not have confidence that long term keys are sufficiently secure. It should be noted that if the long-term key of a VASP becomes compromised then all transactions made with that VASP during the lifetime of that key may be available to an eavesdropper, unless forward secrecy has been provided. Long term keys may have lifetimes of several months, so such an attack could be very attractive to the attacker.

5 Modification of UMTS Security Protocol for Forward Secrecy

We propose two alternatives that are modified from the ASPeCT protocol for UMTS. They require more modulo exponentiations but guarantee forward secrecy. The first protocol is derived from the first prototype (Diffie-Hellman) described above in the paper.

<p>A: user, B: network or VASP</p> <ol style="list-style-type: none"> 1. $A \rightarrow B: g^{r_A}$ 2. $A \leftarrow B: g^{r_B}, h2(K_{AB}, g^{r_B}, B)$ 3. $A \rightarrow B: \{ \{ h3(g^{r_A}, g^b, g^{r_B}, B, K_{AB}) \}^{-1}_{K_A}, A \}^{-1}_{K_{AB}}$ $A, B: K_{AB} = h(g^{r_A r_B}, g^{b r_A})$

Fig. 10. A modified ASPeCT protocol to provide forward secrecy based on the first prototype

The only difference between this modified version and the original ASPeCT protocol is that g^{r_B} is delivered from **B** to **A** instead of r_B , and $g^{r_A r_B}$ is used as a key input to the hash function instead of r_B . The authentic key establishment in the original protocol comes from the fact that (1) from the viewpoint of **B**, a key input g^{r_A} from **A** is included in the signed message of **A** and the fresh nonce r_B of **B**'s own is used for

key computation; (2) from **A**'s viewpoint, the authentic copy of **B**'s private key **b** and **A**'s own fresh nonce r_A are used together to compute the session key; and (3) from any attacker's viewpoint, he cannot compute a key input g^{br_A} with the knowledge of both g^b and g^{r_A} .

Now, the replacement of r_B with g^{r_B} does not affect the protocol with respect to the above three considerations, and helps both principals establish a secret session key satisfying forward secrecy. Note that the compromise of the long-term private key does not lead to the disclosure of the session key due to the inclusion of a Diffie-Hellman form, $g^{r_A r_B}$ in the key computation. The computational cost for forward secrecy in the user (**A**) side is one additional exponentiation to calculate $g^{r_A r_B}$ (still computationally more effective than the STS protocol), and in the network or VASP side, two additional exponentiations to compute $g^{r_A r_B}$ and g^{r_B} (roughly the same computational cost as the STS protocol). The term g^{br_A} is still required in the key computation because this term plays the essential role in authentication of **B** to **A**.

The second protocol is derived from the second prototype for forward secrecy using confidentiality.

A: user, B: network or VASP
1. A \rightarrow B : g^{r_A}
2. A \leftarrow B : $g^{r_A r_B}, h2(K_{AB}, g^{r_A r_B}, B)$
3. A \rightarrow B : $\{ \{ h3(g^{r_A}, g^b, g^{r_A r_B}, B, K_{AB}) \}_{K_A^{-1}}, A \}_{K_{AB}}$
A, B: $K_{AB} = h(g^{r_B}, g^{br_A})$

Fig. 11. Another modified ASPeCT protocol to provide forward secrecy based on the second prototype

In this variant, r_B in the second message and the key computation of the original protocol is replaced by $g^{r_A r_B}$ and g^{r_B} , respectively. This modification does not compromise the protocol with respect to the three considerations as described for the first variant, and enables both principals to establish a secret key satisfying forward secrecy in a different way from the first variant. Retrieval of the term g^{r_B} in the **A** side requires the knowledge of r_A which can be viewed as a private component of the temporary asymmetric pair (r_A, g^{r_A}) . The lifetime of the temporary asymmetric key pairs and r_B should be at most as long as the corresponding session, and hence g^{r_B} and the session key is free from the compromise of the long-term private keys of **A** and **B**. In other words, the secure deletion of the r_A and r_B in both sides after the session thwarts any effort to retrieve the value of g^{r_B} because it was delivered to **A** after encrypted using the temporary public key g^{r_A} of **A**.

Here again, compared to the original ASPeCT protocol, the computational cost for **A** is one more modulo exponentiation to retrieve g^{r_B} from $g^{r_A r_B}$, and for **B**, two more exponentiations to generate g^{r_B} and $g^{r_A r_B}$, which is exactly the same as the first variant for forward secrecy. This clearly shows that the choice of a particular prototype of forward secrecy does not necessarily lead to more expensive or cheaper implementation.

These more expensive versions do not seem likely to find application in the air interface between the user and the network. For this interface is strictly limited by call set-up delay requirements and the possibility of the network private key compromise would be much lower than in the case of VASPs. On the other hand, user to VASP communication looks like a quite feasible target for the protocols because the key compromise problem of VASP would never be negligible and some transaction delay may be considered not so critical in that kind of communications.

Flexibility, or multilevel features, in future mobile security services cannot be over-emphasized considering multimedia features and the complex trust relationships. The alternative protocols described above have the same basic format as the original ASPeCT protocol and hence can be integrated together into a *multilevel featured set* of security protocols.

6 Conclusion

We have surveyed and identified two *general prototypes* for forward secrecy: the first one depending on a particular property of key agreement functions \mathbf{F} and the second one exploiting confidentiality by temporary asymmetric key pairs. We have also applied these two prototypes to the ASPeCT protocol for UMTS and proposed two alternative variations which provide forward secrecy. They require one more modulo exponentiation in the user side and even more computational cost in the network or VASP side when compared to the original ASPeCT protocol. Either of the two modified protocols, however, can be integrated together with the original protocol into a more flexible *set* of protocols which enables a selective protocol usage depending on the required security goals.

References

1. A. Aziz, "SKIP Extension for Perfect Forward Secrecy". Available from <http://www.skip-vpm.org/wetice98/HacknSlash.html>
2. A. Aziz, et al., "Simple Key-Management for Internet Protocols (SKIP)". Available from <http://www.tik.ee.ethz.ch/~skip/SKIP.html>
3. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997, p. 496.
4. C. Günther, "An Identity-based Key-exchange Protocol", *Advances in Cryptology - Eurocrypt'89*, Springer-Verlag, 1990, pp. 29-37.
5. D. P. Jablon, "Strong Password-Only Authenticated Key Exchange", *ACM Computer Communications Review*, October, 1996, pp.5-26.
6. G. Horn and B. Preneel, "Authentication and payment in future mobile systems", *Computer Security - ESORICS'98*, Lecture Notes in Computer Science, 1485, 1998, pp. 277-293.

- 7 . H. Abelson et al., “The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption”, *A Report by an Ad Hoc Group of Cryptographers and Computer Scientists*, 1998. Available from <http://www.cdt.org/crypto/risks98/>
8. IEEE P1363/D9 Standard Specifications For Public Key Cryptography, February. 1999.
9. M. Bellare and S.K. Miner, “A Forward-Secure Digital Signature Scheme”, *Advances in Cryptology - Crypto'99*, Springer-Verlag, 1999.
10. M. Wiener, “Performance Comparison of Public-Key Cryptosystems”, *Cryptobytes*, vol. 1, no. 2, RSA Laboratories, 1998.
11. T. Wu, “The Secure Remote Password Protocol”, *Proceedings of the 1998 Internet Society Network and Distributed Systems Symposium*, pp.97-111. Available from <ftp://srp.stanford.edu/pub/srp/srp.ps>
12. W. Diffie and M. Hellman, “New Directions in Cryptography”, *IEEE Transactions on Information Theory*, Vol. 22, 1976, pp. 644-654.
13. W. Diffie, P. van Oorschot and M. Wiener, “Authentication and Authenticated Key Exchanges”, *Designs, Codes and Cryptography*, 2, 1992, pp. 107-125.

Selecting Cryptographic Key Sizes

Extended Abstract

Arjen K. Lenstra¹, Eric R. Verheul²

¹ Citibank, N.A., 1 North Gate Road, Mendham, NJ 07945-3104, U.S.A.,
arjen.lenstra@citicorp.com

² PricewaterhouseCoopers, GRMS Crypto Group, P.O. Box 85096, 3508 AB Utrecht,
The Netherlands, Eric.Verheul@[nl.pwglobal.com, pobox.com]

Abstract. In this article we give guidelines for the determination of cryptographic key sizes. Our recommendations are based on a set of explicitly formulated hypotheses, combined with existing data points about the cryptosystems. This article is an abbreviated version of [15].

1 Introduction

1.1. Introduction. In this article we offer guidelines for the determination of key sizes for symmetric cryptosystems, RSA, and discrete logarithm based cryptosystems both over finite fields and over groups of elliptic curves over prime fields. Key size recommendations are scattered throughout the cryptographic literature or may be found in vendor documentation. Unfortunately it is often hard to tell on what premises (other than marketability) the recommendations are based. As far as we know [15], of which this is an extended abstract, is the first uniform, clearly defined, and properly documented treatment of this subject for the most important generally accepted cryptosystems. We formulate a set of explicit hypotheses about future developments and apply these uniformly to existing data about the cryptosystems. The resulting key size recommendations are thus obtained in a uniform mechanical way independent of further assumptions or non-scientific considerations. Despite our attempt to be objective we do not expect that our model is to everyone's taste. The underlying model can, however, easily be changed without affecting the overall approach, thereby making this article useful also for those who object to our results.

Our suggestions are based on reasonable extrapolations of developments that have taken place during the last few decades. This approach may fail: a single bright idea may prove that any of the currently popular cryptographic protocols is

Disclaimer. The contents of this article are the sole responsibility of its authors and not of their employers. The authors or their employers do not accept any responsibility for the use of the cryptographic key sizes suggested in this article. The authors do not have any financial or other material interests in the conclusions attained in this paper, nor were they inspired or sponsored by any party with commercial interests in cryptographic key size selection. The data presented in this article were obtained in a two stage approach that was strictly adhered to: formulation of the model and collection of the data points, followed by computation of the lower bounds. No attempt has been made to alter the resulting data so as to better match the authors (and possibly others) expectations or taste. The authors made every attempt to be unbiased as to their choice of favorite cryptosystem, if any. Although the analysis and the resulting guidelines seem to be quite robust, this will no longer be the case if there is some 'off-the-chart' cryptanalytic or computational progress affecting any of the cryptosystems considered here. Indeed, according to at least one of the present authors, strong long-term reliance on any current cryptosystem without very strong physical protection of all keys involved – including public ones – is irresponsible.

considerably less secure than expected. It may even render them completely insecure, as illustrated by the sudden demise of the once popular knapsack-based cryptosystems. In this article we discuss only cryptosystems for which it is believed to be unlikely that such catastrophes will ever occur. For some of these systems non-trivial, but non-catastrophic, new cryptanalytic insights are obtained on a fairly regular basis. So far, a gradual increase in key sizes has been an effective countermeasure against these new insights. It is the purpose of this article to give an idea by how much key sizes have to be increased to maintain a comfortable margin of security.

If sufficiently large quantum computers can be built, then all asymmetric key cryptosystems discussed in this article are insecure (cf. [21]). It is unclear if quantum computers are feasible at all, and our suggestions do not take them into account. Neither do we consider the potential effects of molecular-computing (cf. [19]).

1.2. Run Time Convention. All our run time estimates are based on actual run times or reliable estimates of run times on a 450MHz Pentium II processor. A ‘PC’ always refers to this processor. Computing power is often measured in Mips Years (MY), where a Mips Year is defined as the amount of computation that can be performed in one year by a single DEC VAX 11/780. This measure has often been criticized and we agree with the concerns expressed in [24]. Nevertheless we use MY here as well. We use the convention that one year of computing on a PC is equivalent to 450 MY, but ultimately all our estimates are based on run times on a PC and not on the actual or our definition of MY. The two definitions are, however, sufficiently close (cf. [15]). Our MY figures are therefore compatible with MY figures found elsewhere. We write MMY for one million MY.

1.3. Lower Bounds. Our guidelines are lower bounds in the sense that keys of sizes equal to or larger than the recommended sizes attain at least a certain specified level of security. From a security point of view it is acceptable to err on the conservative side by recommending keys that may be slightly larger than actually required. Most guidelines are therefore obtained by systematically underestimating the effort required for a successful attack. Thus, keys are estimated to be weaker than they are in reality, which is acceptable for our purpose of finding lower bounds. In some cases slight overestimates of the attack effort are used instead, but in those cases there are other factors that ensure that the desired level of security is achieved.

1.4. Equivalence of Attack Efforts. We present key size recommendations for several different cryptosystems. For a certain specified level of security these recommendations may be expected to be equivalent in the sense that the computational effort or number of MY for a successful attack is more or less the same for all cryptosystems under consideration. So, from a computational point of view the different cryptosystems offer more or less equivalent security when the recommended key sizes are used. This *computationally equivalent security* should not be confused with, and is not necessarily the same as, *equipment cost equivalent* security, or *cost equivalent* security for short. We say that two systems offer cost equivalent security if accessing or acquiring the hardware that allows a successful attack in a certain fixed amount of time costs the same amount of dollars for both systems. Note that although the price is the same, the two different attacks may require different hardware.

Following our guidelines does **not** necessarily result in cost equivalent security. In (4.5) we indicate how our guidelines may be changed to obtain cost equivalence, thereby possibly giving up computational equivalence.

The most important reason why we opted for computationally equivalent security as opposed to cost equivalent security is that we found that computational equivalence allows rigorous analysis, mostly independent of our own judgment or preferences. Analysis of cost equivalence, on the other hand, depends on subjective choices that change over time, and that have a considerable effect on the outcome. Thus, for cost equivalence there is a whole spectrum of ‘reasonable’ outcomes, depending on one’s perception of what is reasonable. In (4.5) we present three points of the spectrum.

2 The Cryptographic Primitives

2.1. The Wassenaar Arrangement. The Coordinating Committee for Multilateral Export Controls (COCOM) was an international organization regulating the mutual control of the export of strategic products from member countries to countries that jeopardize their national security. The Wassenaar Arrangement (WA) is a follow-up of the COCOM regulations. In this article we limit ourselves to the 5 types of cryptographic primitives for which a maximum key size that does not require an export license is given in the WA (December 1998, cf. www.wassenaar.org).

We distinguish the cryptographic primitives into symmetric-key (or secret-key) and asymmetric-key (or public-key) cryptosystems and briefly mention cryptographic hash functions as well.

2.2. Symmetric Key Cryptosystems.

Description. In symmetric key cryptosystems the parties share a secret key. The size of the key is its number of bits and depends on the symmetric key cryptosystem.

Wassenaar Arrangement. The maximum symmetric key size allowed by the WA is 56 bits for ‘niche market’ applications and 64 bits for ‘mass market’.

Attacks. Despite many years of research, no method has been published that breaks a DES-encrypted message substantially faster than exhaustive key search, i.e., trying all 2^{56} different keys. The expected number of trials of exhaustive key search is 2^{55} .

Software data points. In 1997 a DES key was successfully retrieved after an Internet search of approximately 4 months (cf. www.rsa.com/des). The expected computing power required for such a software exhaustive key search is underestimated as 0.5 MMY (cf. (1.3)). This estimate is based on the Pentium based figures that a single DES block encryption with a fixed key requires 360 clock cycles or 500 clock cycles with a variable key (cf. [6], [1]). Our estimate lies between two DEC VAX 11/780 estimates that can be found in [7] and [20]. Half a million MY is roughly 13500 months on a PC or 4 months on 3500 PCs, because an exhaustive key search can be evenly divided over any number of processors. For a proper security analysis one therefore has to keep track of the total computational power of the Internet.

Special-purpose hardware data points. At the cost of a one-time investment a hardware attack is substantially faster than a software attack. In 1980 a \$50 million parallel DES key searching machine was proposed with an expected search time of 2

days (cf. [9], [8]), followed in 1993 by a \$1 million, 3½ hour design (cf. [26]). In 1998 a \$130,000, 112 hour machine was built (cf. [13], [11]).

Effectiveness of guessing. There is always the possibility that someone may find a key simply by guessing it. For reasonable key sizes the probability that the correct key is guessed is small: even for a 50-bit key there is a total probability of one in a million that it is found if 10^9 people each make a different guess. With the same effort, the probability of success halves for each additional key bit. Exhaustive key search is nothing more than systematic guessing.

Incomplete attacks. The success probability of exhaustive key search is proportional to the fraction of the key space searched.

Cryptanalytic progress. We assume the existence of a generic symmetric key cryptosystem of arbitrary key size that is about as fast as the DES and for which exhaustive key search is the best attack. Thus, for a b -bit key a successful attack can be expected to require on the order of 2^{b-1} invocations of the underlying function.

2.3. Asymmetric Key Cryptosystems. If the private key of an asymmetric key cryptosystem can be derived from the public key, then the system can be broken. What the keys consist of, and how hard it is to break the system, depends on the type of asymmetric key cryptosystem. We distinguish the following three types:

1. Classical asymmetric systems;
2. Subgroup discrete logarithm systems;
3. Elliptic curve systems.

2.3.1. Classical Asymmetric Systems. These refer to RSA and traditional discrete logarithm (TDL) systems, such as the Diffie-Hellman scheme and ElGamal systems.

RSA description. In RSA the public key contains a large non-prime number, the RSA modulus, which is chosen as the product of two large primes. The security of RSA is based on the difficulty of the integer factorization problem. The size of an RSA key refers to the bit-length of the RSA modulus. This should not be confused with the number of bits required to store an RSA public key, which is usually slightly more.

TDL description. In a TDL system the public key consists of a finite field F_p of size p , a generator g of the multiplicative group $(F_p)^*$ of F_p , and an element y of $(F_p)^*$ that is not equal to 1. We assume that the field size p is such that $p-1$ has a prime factor of roughly the same order of magnitude as p . The private key is the discrete logarithm of y with respect to g , i.e., the smallest positive integer m such that $g^m = y$. The private key m is at least 1 and at most $p-2$. The security of TDL systems is based on the difficulty of computing discrete logarithms in the multiplicative group of a finite field. The size of a TDL key refers to the bit-length of the field size p . The number of bits required to store a TDL public key is larger, since it contains g and y as well.

Wassenaar Arrangement. Both the maximal RSA modulus size and the maximal field size allowed by the WA are 512 bits.

Attacks. Factoring an RSA-modulus n by exhaustive search amounts to trying all primes up to \sqrt{n} . Finding a discrete logarithm by exhaustive search requires on the order of p operations in F_p . Thus, if exhaustive search were the best attack on these systems, then 112-bit RSA moduli or 56-bit p 's would give security comparable to the DES. However, there are much more efficient attacks and much larger keys are

required. The methods to attack these two entirely different problems are similar, which is why we treat RSA and TDL systems as the same category.

The fastest factoring algorithm published today is the Number Field Sieve (NFS), which is based on an idea by John Pollard. On heuristic grounds NFS can be expected to require time proportional to

$$L[n] = e^{(1.9229 + o(1)) * \ln(n)^{1/3} * \ln(\ln(n))^{2/3}}$$

to factor an RSA modulus n , where the $o(1)$ term goes to zero as n goes to infinity. This run time is called *subexponential* in n because as n goes to infinity it is less than n^c for any $c > 0$. The storage requirements of the NFS are proportional to $\sqrt{L[n]}$. If p is a prime number then a discrete logarithm variation of the NFS (DLNFS) finds a discrete logarithm in F_p in expected time proportional to $L[p]$.

These run time estimates cannot be used directly to estimate the number of operations required to factor a certain n or to compute discrete logarithms in a certain F_p . For n and p of about the same size, $L[n]$ and $L[p]$ are approximately equal if the $o(1)$'s are omitted, but the discrete logarithm problem in F_p is considerably more difficult than factoring n . As shown by extensive experiments the estimates can be used for limited range extrapolation. If one knows, by experimentation, that factoring an RSA modulus n using NFS takes time t , then factoring some other RSA modulus $m > n$ will take time close to $t*L[m]/L[n]$ (omitting the $o(1)$'s), if the sizes of n and m do not differ by too much. If, however, m is much bigger than n , then the effect of the $o(1)$ going to zero can no longer be ignored (cf. [23]), and $t*L[m]/L[n]$ will be an overestimate of the time to factor m . The same run time extrapolation method applies to the DLNFS.

Software data points. The largest published factorization using the NFS is that of the 512-bit number RSA155, an RSA modulus of 155 decimal digits (cf. [5]). This effort was estimated to cost at most 20 years on a PC with at least 64Mbytes of memory (or a single day on 7500 PCs). It is less than 10^4 MY and corresponds to fewer than $3*10^{17}$ operations, whereas $L[10^{155}] = 2*10^{19}$ (omitting the $o(1)$). This shows that $L[n]$ overestimates the number of operations to be carried out for the factorization of n . The run time given here is the actual run time of the RSA155 factoring effort and should not be confused with the estimates given in [24] which appeared around the same time and which are 100 times too high (cf. [17]). This run time is only a fraction of the cost of a software DES key search, but the NFS requires much more memory.

Practical experience with the DLNFS is still limited. It is generally accepted that, for any b in the current range of interest, factoring b -bit integers takes about the same amount of time as computing discrete logarithms in $(b-x)$ -bit fields, where x is a small constant around 20. Below we do not present key size suggestions for TDL systems and recommend using the RSA key size suggestions for TDL systems as well.

Special-purpose hardware data points. Special-purpose hardware devices are occasionally proposed for factoring algorithms, but no useful data points have been published. Due to the complexity of the underlying factorization algorithms and the corresponding hardware design it is for any special-purpose hardware factoring device difficult to achieve parallelization at a reasonable cost and at a scale comparable to hardware attacks on the DES, but it may not be impossible. Given the current state of the art we consider it to be unlikely that special-purpose hardware will

have a noticeable impact on the security of RSA moduli. But we find it imprudent to ignore the possibility altogether, and warn against too strong reliance on the belief that special-purpose attacks on RSA are impossible. To illustrate this, the quadratic sieve factoring method was implemented successfully on a Single-Instruction-Multiple-Data architecture (cf. [10]). A SIMD machine is by no means special-purpose hardware, but it could be relatively cheap compared to ordinary PCs.

Effectiveness of guessing. Key sizes for classical asymmetric systems have to be larger than 512 to obtain any security at all. Breaking the system by guesswork is thus out of the question. So, from this point of view, classical asymmetric systems seem to be more secure than symmetric key cryptosystems. For RSA there is more to this story, as shown below.

Incomplete attacks. Both the NFS and the DLNFS are effective only if run to completion. RSA, however, can be attacked also by the Elliptic Curve Method (ECM). After a relatively small amount of work this method produces a factor with much higher probability than mere guesswork: if one billion people were to attack a 512-bit RSA modulus, each by running the ECM for just one hour on their PC, then the probability that one of them would factor the modulus is more than 10%. For a 768-bit RSA modulus the probability of success of the same computational effort is about one in a million. Admittedly, this is a very low success probability for a tremendous effort – but the success probability is orders of magnitude larger than guessing, while the amount of work is of the same order of magnitude. No discrete logarithm equivalent of the ECM has been published. See also (5.9).

Cryptanalytic progress. Classical asymmetric systems are the prime example of systems for which the effectiveness of cryptanalysis is steadily improving. The current state of the art of factoring (and discrete logarithm) algorithms should not be interpreted as the culmination of many years of research but is just a snapshot of work in progress. We illustrate this point with a list of some of the developments since the early seventies, each of which had a substantial effect on the difficulty of factoring or computing discrete logarithms: continued fraction method, linear sieve, quadratic sieve, multiple polynomial variation, Gaussian integers, loosely coupled parallelization, multiple large primes, special number field sieve, structured Gaussian elimination, number field sieve, singular integers, lattice sieving, block Lanczos or conjugate gradient, and sieving-based polynomial selection for NFS. We assume that this trend of continuous algorithmic developments will continue in the years to come.

It has never been proved that breaking RSA is equivalent to factoring the RSA modulus. Indeed, for RSA there is evidence that the equivalence does not hold if the public exponent is small. We therefore explicitly assume that breaking RSA is equivalent to factoring the RSA modulus. In particular, we assume that the public exponent for RSA is sufficiently large. Furthermore we restrict ourselves to TDL based protocols for which attacks are provably equivalent to either computing discrete logarithms or solving the Diffie-Hellman problem. There is strong evidence that the latter problem is equivalent to computing discrete logarithms

2.3.2. Subgroup Discrete Logarithm Systems.

Description. Subgroup discrete logarithm (SDL) systems are like traditional discrete logarithm systems, except that g generates a relatively small, but sufficiently large, subgroup of the multiplicative group $(\mathbb{F}_p)^*$. The size of the subgroup is prime and is

indicated by q . The private key m is at least 1 and at most $q-1$. The security of SDL is based on the difficulty of computing discrete logarithms in a subgroup of the multiplicative group of a finite field. These can be computed if discrete logarithms in the full multiplicative group can be computed. Therefore, the security of an SDL system relies on the sizes of both q and p . Nevertheless, the size of an SDL key simply refers to the bit-length of the subgroup size q , where the field size p is given by the context. The actual number of bits required to store an SDL public key is substantially larger than the SDL key size q , since the public key contains p , g and y as well.

Wassenaar Arrangement. The maximum SDL field size allowed by the WA is 512 bits – there is no maximum allowed key size. A popular subgroup size is 160 bits.

Attacks. Methods that can be used to attack TDL systems also can be used to attack SDL systems. The field size p should therefore satisfy the same security requirements as in TDL systems. But the SDL problem can also be attacked directly by Pollard's rho method, which dates from 1978, and by Shanks' even older baby-step-giant-step method. These methods can be applied to any group if the group elements allow a unique representation and the group law can be applied efficiently – unlike the DLNFS it does not rely on any special properties that group element representations may have. The expected run time of Pollard's rho method is *exponential* in q , namely $1.25\sqrt{q}$ group operations, i.e., multiplications in \mathbb{F}_p . Its storage requirements are very small. Shanks' method needs about the same number of operations but needs storage for about \sqrt{q} group elements. Pollard's rho method can easily be parallelized over any number of processors resulting in a linear speedup (cf. [25]). Furthermore, there is no post-processing involved in Pollard's rho (unlike the (DL)NFS, where after completion of the first step a cumbersome matrix step has to be carried out), although for the parallelized version substantial amounts of storage space should be available.

Data points. We have not been able to find any useful data about the effectiveness of the parallelized Pollard rho attack on SDL systems. Our figures below are based on an adaptation of data points for elliptic curve systems, cf. (4.1).

Effectiveness of guessing. As long as SDL keys are not shorter than 112 bits (permitted by the WA for EC systems, see below), guessing the private key requires guessing at least 112 bits. This may safely be assumed to be infeasible.

Incomplete attacks. The success probability of Pollard's rho method is, roughly speaking, proportional to the square of the fraction of the work performed, i.e., for any x , $0 \leq x \leq 1$, the chance is x^2 that the key is found after performing a fraction x of the expected $1.25\sqrt{q}$ group operations.

Cryptanalytic progress. Since the invention of Pollard's rho method in 1978 no new results have been obtained that threaten SDL systems, with the exception of the efficient parallelization of Pollard's rho method in 1996. The only reasonable extrapolation of this rate of progress is to assume that no substantial progress will be made. The results in [18, 22] that, in a certain generic model of computation, Pollard's rho is essentially the best one can do may be comforting in this context. It should be kept in mind, however, that the generic model does not apply to any practical situation that we are aware of, and that the possibility of a subexponential attack against SDL systems cannot be ruled out.

2.3.3. Elliptic Curve Systems.

Description. Elliptic curve (EC) systems are like SDL systems, except that g generates a subgroup of the group H of points on an elliptic curve E over a finite field F_p . The size q of the subgroup generated by g is prime and the private key m is in the range $[1, q-1]$. The security of EC systems is based on the difficulty of computing discrete logarithms in a subgroup of H . These can be computed if discrete logarithms in H can be computed. This problem is known as the ECDL problem. No better method to solve the ECDL problem is known than by solving the problem in all cyclic subgroups and by combining the results. The difficulty of the ECDL problem therefore depends on the size of the largest prime divisor of the order of H (which is close to p). For that reason, p , E , and q are usually chosen such that the sizes of p and q are close. Thus, the security of EC systems relies on the size of q , and the size of an EC key refers to the bit-length of the subgroup size q . The actual number of bits required to store an EC public key may be substantially larger than the EC key size q , since the public key contains p , E , g , and y as well.

Wassenaar Arrangement. The maximum EC key size allowed by the WA is 112 bits, with unspecified field size. For prime fields a popular size is 160 bits both for the field and subgroup size. For non-prime fields a popular choice is $p = 2^{163}$ with a 161-bit q .

Attacks. A DL NFS equivalent or other subexponential method to attack EC systems has never been published. The most efficient method published to attack EC systems is Pollard's parallelizable rho method, with an expected run time of $0.88\sqrt{q}$ group operations. The number of field multiplications per group operation is about 12.

Software data points. The cost of the group operation is proportional to $(\log_2(q))^2$. From the estimates given on www.certicom.com/chal we derive that a 109-bit EC system with $p = 2^{109}$ should take about 18,000 years on a PC (or, equivalently, one year on 18,000 PCs) which is about 8 MMY. This computation is feasible on a large network of computers. It also follows from www.certicom.com/chal that an attack on a 109-bit EC system with a prime p of about 109 bits should take about 2.2 MMY. This is an underestimate because it is based on primes of a special form (cf. [12]). Nevertheless, it is used as the basis for extrapolations to estimate the effort required for software attacks on larger EC systems over prime fields (cf. (1.3)).

Special-purpose hardware data points. In 1996 an attack against a 120-bit EC system with $p = 2^{155}$ was sketched (and published 3 years later, cf. [25]). Building this design would cost \$10 million and it would take about 32 days. The designers claim that an attacker can do better by using current silicon technology and that further optimization may be obtained from pipelining. This is further discussed in (3.6).

Effectiveness of guessing. As long as EC keys are not shorter than the 112 bits permitted by the WA, guessing the private key requires guessing at least 112 bits which may safely be assumed to be infeasible.

Incomplete attacks. As with Pollard's rho attack against SDL systems its success probability is proportional to the square of the fraction of the work performed.

Cryptanalytic progress. The remarks made above on SDL systems apply here as well. It is therefore not unreasonable to base our figures below on the assumption that there will be no substantial progress in the years to come. For EC systems this is not something we feel comfortable with, because EC related cryptanalytic results are obtained quite regularly. So far, most of these results affected only special cases. We

therefore make the explicit assumption that curves are picked at random and that only curves over prime fields are used. Even then, it is not hard to find researchers who believe that the rich mathematical structure of elliptic curves may still have some surprises in store. Others argue that the ECDL problem has been studied extensively, and that EC systems are sufficiently secure. We do not want to take a position in this argument and we simply suggest two key sizes for EC systems: one based on ‘no cryptanalytic progress’ and one based on ‘cryptanalytic progress at the same rate as for RSA and TDL systems’. The reader may then interpolate between the two types of extrapolations according to her own taste.

2.4. Cryptographic Hash Functions.

Description. A cryptographic hash function is a function that maps an arbitrary length message to a fixed length ‘hash’, satisfying various properties that are beyond the scope of this article. The size of the hash function is the length in bits of the hash.

Attacks. Cryptographic hash functions can be attacked by the birthday paradox attack. The number of hash function applications required by a successful attack is expected to be proportional to $2^{x/2}$, where x is the size of the hash function. We assume that cryptographic hash functions have to be ‘any collision-resistant’. For ‘target collision-resistant’ hashes the sizes may be halved assuming the hash function is properly used.

Software data points. In [3] 241, 345, 837, and 1016 Pentium cycles are reported for MD4, MD5, SHA-1, and RIPEMD-160, respectively. Thus, the software speed of a hash function application as used by a birthday paradox attack is comparable to the software speed of a single DES block encryption (cf. (2.2)).

Special-purpose hardware data points. Special-purpose hardware has been designed for several hash functions. We may assume that their speed is comparable to the speed of special-purpose exhaustive key search hardware.

Cryptanalytic progress. We assume the existence of a generic cryptographic hash function of speed comparable to the existing functions mentioned above and for which the birthday paradox attack is the best attack. It follows that an attack on our generic symmetric key cryptosystem of key size b can be expected to take about the same time as an attack on our generic cryptographic hash function of size $2b$. Thus, a lower bound for the size of the latter follows by doubling the lower bound for the size of symmetric key cryptosystems. Because of this simple ‘rule of thumb’, sizes of cryptographic hash functions are not discussed in the sequel.

3 The Model

3.1. Key Points. The choice of cryptographic key sizes depends primarily on the following four points:

- I. Life span: the expected time the information needs to be protected.
- II. Security margin: an acceptable degree of infeasibility of a successful attack.
- III. Computing environment: the expected change in computational resources available to attackers.
- IV. Cryptanalysis: the expected developments in cryptanalysis.

Efficiency and storage considerations may also influence the choice of key sizes, but since they are not directly security-related they are not discussed here.

3.2. Life Span. In the table in Section 4 key sizes are suggested, depending on the expected life span of the cryptographic application. It is the user's responsibility to decide until what year the protection should be effective.

3.3. Security Margin. A cryptosystem can be assumed to be secure only if it is considered to be sufficiently infeasible to mount a successful attack. It is hard to quantify what this means precisely. One could, for instance, decide that a key size for a certain cryptosystem is secure if breaking it would be, say, 10^6 times harder than the largest key size that can currently be broken. There are several problems with this approach. First of all, the choice 10^6 is rather arbitrary. Secondly, there is no reason to believe that the ‘largest key broken so far’ accurately represents the best that can currently be done. In the third place, for some of the cryptographic primitives considered here data may not be available or they may be outdated (SDL, TDL), thereby ruling out uniform application of this approach. We opt for a different approach.

Hypothesis I. As the basis for our extrapolations we assume that the DES was at least sufficiently secure for commercial applications until 1982 because it was introduced in 1977 and stipulated to be reviewed every five years. We therefore hypothesize that in 1982 a computational effort of 0.5 MMY was believed to provide an adequate security margin for commercial DES applications against software attacks (cf. (2.2)). As far as hardware attacks are concerned, we assume that the “\$50 million, 2 days” DES key searching machine (cf. (2.2)) from 1980 was not considered to be a serious threat for commercial applications of the DES at least until 1982. We stress ‘commercial applications’ because, even for 1980 budgets, \$50 million and 2 days are not an insurmountable obstacle for certain organizations. Our hypothesis is further discussed below (cf. (3.8)). We note that quite different assumptions allow an approach similar to ours, though the resulting guidelines will be different (cf. (4.4)).

3.4. Computing Environment.

Hypothesis II. To estimate how the computing power available to attackers may change over time we use a variation of Moore's law. Moore's law states that the density of components per integrated circuit doubles every 18 months. A widely accepted interpretation of this law is that the computing power per chip doubles every 18 months. There is some skepticism whether this law will, or even can, hold much longer. Therefore we hypothesize a less technology dependent variation that so far seems to be sufficiently accurate: every 18 months the amount of computing power and random access memory one gets for a dollar doubles. Thus, for the same cost one gets a factor of $2^{10*12/18} \approx 100$ more computing power and fast memory every 10 years, either in software on multipurpose chips (PCs) or using special-purpose hardware.

To illustrate this, it is not unreasonable to assume that a cheaper and slower version of the 1980 “\$50 million, 2 days” DES key searching machine would be a “\$1 million, 100 days” machine, i.e., 50 times less hardware and therefore 50 times slower. According to our version of Moore's law the \$1 million machine may be expected to be $2^{8.7}$ times faster in 1993, since there are $12*13 = 18*8.66$ months between 1980 and 1993. Since $2^{8.7} \approx 406$ the 1993 version would need about $100/406$ days, i.e., about 6 hours, which is indeed close to the $3\frac{1}{2}$ hours required by the \$1

million design from [26]. On the other hand, further extrapolation suggests that in 1998 a \$1 million machine may be expected to take 0.6 hours, or that a \$130,000 machine would take 4.6 hours, i.e., about 24 times faster than the machine that was actually built in 1998 (cf. [13]). This anomaly is due to the fact that building the \$130,000 machine was, relatively speaking, a small scale enterprise where every doubling of the budget would have quadrupled the performance (cf. [14]).

Hypothesis III. Our version of Moore's law implies that we have to consider how budgets may change over time. The US Gross National Product shows a trend of doubling every ten years: \$1630 billion in 1975 measured in 1975 dollars, \$4180 billion in 1985 measured in 1985 dollars, and \$7269 billion in 1995 in 1995 \$'s. This leads to the hypothesis that the budgets of organizations doubles every ten years.

Combination of Hypotheses I, II, and III. If in 1982 an amount of computing power of 0.5 MMY is assumed to be infeasible to invest in an attack, then 100 ($\approx 2 \cdot 100 \cdot 0.5$) MMY is infeasible in 1992. Furthermore, $2 \cdot 10^4$ ($\approx 200 \cdot 100$) MMY is infeasible in 2002, and $4 \cdot 10^6$ MMY is infeasible in 2012.

3.5. Cryptanalysis.

Hypothesis IV. It is impossible to say what cryptanalytic developments will take place, or have already taken place surreptitiously. We find it reasonable to assume that the pace of (published) future cryptanalytic findings and their impact are not going to vary dramatically compared to what we have seen from 1970 until 1999. For classical asymmetric systems the effect of cryptanalytic developments illustrated in (2.3) is similar to Moore's law, i.e., 18 months from now we may expect that attacking the same classical asymmetric system costs half the computational effort it costs today, cf. (4.2). For all other systems we assume that no substantial cryptanalytic developments will take place, with the exception of elliptic curve systems for which we use two types of extrapolations: no progress and progress à la Moore.

3.6. Software versus Special-Purpose Hardware Attacks. The proposed key sizes in the next section are obtained by combining Hypotheses I-IV with the software based MY data points. This implies that all extrapolations are based on 'software only' attacks and result in computationally equivalent key sizes (cf. (1.4)). One may object that this does not take special-purpose hardware attacks into account. Here we discuss to what extent this is a reasonable decision, and how our results should be interpreted to take special-purpose hardware attacks into account as well.

Symmetric key systems. In 1980 the DES could either be broken at the cost of 0.5 MMY, or using a "\$50 million, 2 days" machine. This is consistent with our version of Moore's law and the 1993 design from [26]. Thus, it seems reasonable to assume that a DES attack of one MMY is comparable to an attack by [\$10 million, 20 days, 1980]-hardware or, using Moore's law, by $[\$200/2^{10.66} \text{ million} = \$125,000, 1 \text{ day}, 1996]$ -hardware. It also follows that the 1982 relation between software and special-purpose hardware attacks on the DES has not changed. Thus, if one assumes that the DES was sufficiently resistant against a special-purpose hardware attack in 1982, the same holds for the symmetric key sizes suggested for the future, even though they are based on extrapolations of 'software only' attacks. Our estimates and the resulting cost of special hardware designs are consistent with the estimates given in [2] and [4].

EC systems. The cost of a software attack on a 109-bit EC system with $p = 2^{109}$ was

estimated as 8 MMY, so that attacking a 120-bit EC system with $p = 2^{155}$ should take about $(2^{(120-109)/2})*(155/109)^2 \approx 91$ times longer, i.e., about 730 MMY. The [\$10 million, 32 days, 1996]-hardware design attacking a 120-bit EC system with $p = 2^{155}$ (cf. (2.3.3)) should thus be comparable to 730 MMY. However, that design was based on 1992 technology which can be improved by using 1996 technology. So, by Moore's law, the 'upgraded' [\$10 million, 32 days, 1996]-hardware design could be comparable with $730*6.35 \approx 4600$ MMY. It follows that an EC attack of one MMY is comparable to [\$70,000, 1 day, 1996]-hardware.

We find that one MMY is equivalent to [\$70,000 to \$125,000, 1 day, 1996]-hardware. Thus, it is tempting to suggest that one MMY is approximately equivalent to [\$10⁵, 1 day, 1996]-hardware; more generally, that one MMY would be equivalent to $[\$10^5/2^{2*(y-1996)/3}, 1 \text{ day}, y]$ -hardware in year y . This conversion formula would allow us to go back and forth between software and special-purpose hardware attacks, and make our entire model applicable to hardware attacks as well.

In our opinion the consistency between the two conversions is a mere coincidence. In the first place, the estimate holds only for relatively simple minded DES or EC cracking devices for EC systems over non-prime fields (i.e., those with $p = 2^k$), not for EC systems over prime fields or full-blown PCs. For prime fields the hardware would be slower, whereas in software EC systems can be attacked faster over prime fields than over non-prime fields (cf. (2.3.3)). Thus, for special-purpose hardware attacks on EC systems over prime fields the consistency no longer holds. Secondly, the pipelined version of the EC-attacking special-purpose hardware from [25] would be about 7 times faster (cf. [27]), so that also for special-purpose hardware attacks on EC systems over non-prime fields the consistency between DES and EC attacks is lost. The prime field version of the pipelined device would be 2^4 to 2^5 times slower than the non-prime field version (cf. [27]). The details of the pipelined device have not been published, cf. [28].

As mentioned in (2.3.3), we consider only EC systems that use randomly selected curves over prime fields. We show that we may base our recommendations on 'software only' attacks, if we use the software based data point that a 109-bit EC system can be attacked in 2.2 MMY (cf. (2.3.3)). The 2.2 MMY underestimates the true cost, and is lower than the 8 MMY cost to attack the non-prime field of equivalent size. The latter can be done using non-pipelined special-purpose hardware in a way that is consistent with our DES infeasibility assumption, as argued above. For special-purpose hardware a non-prime field can be attacked faster than a prime field of equivalent size, so if we use the naive DES-consistent hardware conversion, then the hypothetical special-purpose hardware that follows from extrapolation of the 2.2 MMY figure to larger prime fields underestimates the true hardware cost. That means that the resulting key sizes are going to be too large, which is acceptable since we are deriving lower bounds (cf. (1.3)). The more realistic prime field equivalent of the non-DES-consistent pipelined device for non-prime fields is, based on the figures given above, at least $2^4*8/(2.2*7) > 8$ times slower than our hypothetical hardware. This implies that the more realistic hardware would lead to lower key sizes than the hypothetical hardware. Thus, it is acceptable to stick to the latter (cf. (1.3)). It follows that, if one assumes that the DES was sufficiently resistant against a special-purpose

hardware attack in 1982, the same holds for the EC key sizes suggested for the future, even though they are based on extrapolations of ‘software only’ attacks.

SDL systems. The same holds for SDL systems because our analysis of SDL key sizes is based on the EC analysis as described below.

Classical asymmetric systems. For classical asymmetric systems we do not consider special-purpose hardware attacks, as argued in (2.3.1). The issue of software attacks on classical asymmetric systems versus special-purpose hardware attacks on other cryptosystems is discussed below.

Equipment cost comparison of software and special-purpose hardware attacks. Our recommendations below are computationally equivalent and, as argued above, they all offer security at least equivalent to the 1982 security of the DES, both against software and special-purpose hardware attacks. That does not necessarily imply that the key sizes for the various cryptosystems are also cost equivalent, because the equipment costs of the 1982 software and special-purpose hardware attacks on the DES are not necessarily equal either. One point of view is that accessing the hardware required for software attacks is for free, as in all Internet based cryptosystem attacks so far and other large computational Internet projects. Adoption of this rule would make computational and cost equivalence identical, which is not generally acceptable (cf. [27]). A precise equipment cost defies exact analysis, primarily because no precise ‘cost of a PC’ can be pinpointed. Nevertheless, we sketch how an analysis based on cost equivalence could be carried out.

According to newspaper advertisements fully equipped PCs (cf. (1.2)) can be bought for prices varying from \$0 to \$450. The ‘free’ machines support the point of view that software attacks are for free. Assume that one does not want to deal with the strings attached to the free machines and that a stripped down PC (i.e., a 450 MHz Pentium II processor, a mother-board, and communications hardware) costs \$100. It follows that [\$81 million, 1 day, 1999]-hardware is equivalent to at least one million software MY, disregarding the possibly much larger quantum discount one should be able to negotiate for an order of this size. Compared to the above exhaustive key search [\$125,000, 1 day, 1996] \approx [\$31,000, 1 day, 1999]-hardware, software MY are thus about 2500 times more expensive. Compared to the pipelined [\$70,000/7, 1 day, 1996] \approx [\$2500, 1 day, 1999]-hardware to attack EC systems over non-prime fields, software MY are more than 3×10^4 times more expensive, but at most about 2×10^3 times more expensive than the prime field version of the pipelined design.

It follows that for our purposes software MY are at most 2500 times more expensive than MY produced by special-purpose hardware. In (4.5) it is shown how this factor 2500 can be used to derive equipment cost equivalent key sizes from the computationally equivalent ones. The factor 2500 should be taken with a grain of salt. Its scientific merit is in our opinion questionable because it is based on a guess for the price of stripped down PCs and the presumed infeasibility of special-purpose hardware attacks on RSA (cf. (2.3.1) and the pipelined design in [10]).

3.7. Memory Considerations. In [15] we explain why (NFS) memory requirements do not explicitly have to be taken into account when extrapolating run times.

3.8. Remark. We do not expect that everyone agrees with our hypotheses. In particular Hypothesis I is debatable. Note that we did not assume anything about the (un)breakability of the DES in any year. We assumed that it offered enough security

for commercial applications, not that well-funded government agencies were unable to break it back in 1977. In this context it may be entertaining to mention that Mike Wiener, after presenting his [\$1 million, 3½ hours, 1993]-hardware design at a cryptography conference, was told that he had done a nice piece of work and he was offered a similar machine at only 85% of the cost – with the catch that it was 5 years old (cf. [28]). Anyone who prefers a stronger or weaker infeasibility assumption can still use our approach, as shown in (4.4). Also our Hypothesis II is certainly not to everyone’s taste. Some argue that Moore’s law cannot hold much longer, others (cf. [14]) find it too pessimistic. Hypothesis II thus represents a reasonable compromise.

4 Lower Bound Estimates for Cryptographic Key Sizes

4.1. Method of Computation. For year y we first compute $\text{IMY}(y)$, the number of MY considered to be infeasible for that year, based on Hypotheses I-III:

$$\text{IMY}(y) = 0.5 * 10^6 * 2^{2(y-1982)/3} * 2^{(y-1982)/10}.$$

The resulting value is used to derive key sizes that should be sufficiently secure until year y , for all cryptographic primitives considered in Section 2. For symmetric key cryptosystems the key size is computed as the smallest integer that is at least

$$56 + \log_2(\text{IMY}(y) / (0.5 * 10^6)) = 23y / 30 - 1463.533.$$

For classical asymmetric systems we use the asymptotic run time $L[n]$ of the NFS (omitting the $o(1)$), the data point that a 512-bit key was broken in 1999 at the cost of less than 10^4 MY (cf. (2.3.1)) and Hypothesis IV that cryptanalytic progress à la Moore is expected, and we determine the smallest size s such that

$$L[2^s] * 10^4 \geq L[2^{512}] * 2^{2(y-1999)/3} * \text{IMY}(y).$$

Because the data point used overestimates the cost of factoring a 512-bit key and because we omit the $o(1)$ the difficulty of breaking classical asymmetric systems with keys of size s is overestimated (cf. (2.3.1)), i.e., the RSA and TDL key sizes should be even larger than given in Table 1.

For SDL systems we use the just determined s as field size for year y . Because no suitable data points are available, we use the optimistic estimate that an EC system over a prime field of 109 bits can be broken in 2.2 MMY (cf. (2.3.3)) and that an elliptic curve operation takes on average 9 field multiplications. Combined with the relative speed of Pollard’s rho method, and the expected growth of the cost of the field operations, we find that the size of the subgroup size q can be taken as

$$109 + 2 * \log_2(109^2 * \text{IMY}(y) * 9 / (s^2 * \sqrt{2} * 2.2 * 10^6)).$$

The resulting sizes are at most two bits too large (cf. (1.3) and [15]).

For EC systems we use the same optimistic estimate that a 109-bit system can be broken in 2.2 MMY combined with the expected growth rate of the number of group operations required by Pollard’s rho method and the expected growth of the cost of the group operations, to determine the smallest size t such that

$$t^2 * 2^{(t-109)/2} \geq 109^2 * \text{IMY}(y) / (2.2 * 10^6).$$

The resulting t represents the recommended EC key size lower bound if no cryptanalytic progress is assumed, where it should be noted that t is on the large side because the 2.2 MMY estimate is rather optimistic (cf. (1.3)). An alternative key size that assumes cryptanalytic progress à la Moore is found by taking the smallest u with

$$u^2 * 2^{(u-109)/2} \geq 2^{2(y-1999)/3} * 109^2 * \text{IMY}(y) / (2.2 * 10^6).$$

4.2. Remarks on the Computation of Table 1.

1. All estimates may be computed for years before 1999. Some of the resulting data can be found in Table 1. Strictly speaking this does not make sense for the “EC with progress” column. It is described in (4.4) how to use the data in italics.
2. The results do not change significantly if the RSA data point is replaced by other (older) factoring data points, which validates our decision to adopt a Moore-like law for cryptanalytic progress affecting classical asymmetric systems.

4.3. Using Table 1. Assuming one agrees with our hypotheses, Table 1 can be used as follows. Suppose electronic information has to be guaranteed until the year 2020. Looking at the row for the year 2020 in Table 1, one finds that an amount of computing of $2.9 * 10^{14}$ MY in the year 2020 may be considered to be as infeasible as $0.5 * 10^6$ MY was in 1982. Security comparable to the security offered by the DES in 1982 is therefore obtained by using, in the year 2020:

- Symmetric keys of **at least** 86 bits, and hash functions of **at least** 172 bits;
- RSA moduli of **at least** 1881 bits; the meaning of the ‘1472’ given in the second entry of same column is explained in 4.5
- SDL systems with subgroups of **at least** 151 bits over fields of **at least** 1881 bits.
- EC systems over prime fields of **at least** 161 bits if one trusts that no cryptanalytic progress will occur, **at least** 188 bits if one wants to be more careful.

If finite fields are used that allow faster operations than suggested by our estimates, the SDL and EC data in Table 1 can still be used: if the arithmetic goes x times faster, keys should be roughly $2 * \log_2(x)$ bits larger than indicated. As noted above the field arithmetic is already assumed to be quite fast. Similarly, if one finds that the data point used for EC systems overestimates the cost by a factor x , i.e., that the 2.2 MMY to attack 109-bit EC systems should be only $2.2/x$ MMY, add roughly $2 * \log_2(x)$ bits to the suggested EC key sizes.

4.4. Alternative Security Margin. For corporations that have used the DES beyond 1982 our infeasibility assumption of 0.5 MMY in 1982 may be too strong. For others it may be too weak. Here we explain how to use Table 1 to look up key sizes for year y , for example $y = 2005$, if one trusts the DES until the year $1982 + x$, where x is negative if our infeasibility assumption is considered to be too weak and positive otherwise. So, for example, $x = 13$ if one trusts the DES until 1995.

- Symmetric keys: take the entry for year $y - x$, i.e., $2005 - 13 = 1992$ in our example. The resulting symmetric key size suggestion is 64 bits.
- Classical asymmetric keys: take the entry for year $y - 23*x/43$, i.e., $2005 - 23*13/43 \approx 1998$ in our example. So 879-bit RSA and TDL keys should be used.

- SDL keys: take the classical asymmetric key size s' for year $y - 23*x/43$, the SDL size t for year $y - x$, and the classical asymmetric key size s for year $y - x$ and use a subgroup of size $t + 4*\log_2(s) - 4*\log_2(s')$ over a field of size s' . In our example $s' = 879$, $t = 114$, and $s = 682$ so that a subgroup of size $114 + 4*\log_2(682) - 4*\log_2(879) \approx 113$ bits should be used with a 879-bit field.
- EC systems without progress: take the ‘without progress’ entry for year $y - x$, i.e., $2005 - 13 = 1992$ in the example. The resulting EC key size suggestion is 120 bits.
- EC systems with progress à la Moore: take the ‘with progress’ entry for year $y - 23*x/43$, i.e., $2005 - 23*13/43 \approx 1998$ in our example. The resulting EC key size suggestion is 129 bits.

The Table 1 entries in italics for years before 1999 may be used in the last application; the other italics entries may be used if $x < 0$.

Table 1

Lower bounds for computationally equivalent key sizes,
assuming cryptanalytic progress à la Moore affecting classical asymmetric systems

Year	Symmetric Key Size	Classical Asymmetric Key Size (and SDL Field Size)	Subgroup Discrete Logarithm Key Size	Elliptic Curve Key Size		Infeasible number of Mips Years	Lower bound for Hardware cost in US \$ for a 1 day attack (cf. (4.5))	Corresponding number of years on 450MHz PentiumII PC			
				Progress							
				no	yes						
1982	56	417 <small>288</small>	102	105	85	$5.00 * 10^5$	$3.98 * 10^7$	$1.11 * 10^3$			
1983	57	440 <small>288</small>	103	107	88	$8.51 * 10^5$	$4.27 * 10^7$	$1.89 * 10^3$			
1984	58	463 <small>320</small>	105	108	89	$1.45 * 10^6$	$4.57 * 10^7$	$3.22 * 10^3$			
1985	59	488 <small>320</small>	106	110	93	$2.46 * 10^6$	$4.90 * 10^7$	$5.47 * 10^3$			
1986	60	513 <small>352</small>	107	111	96	$4.19 * 10^6$	$5.25 * 10^7$	$9.31 * 10^3$			
1987	60	539 <small>384</small>	108	113	98	$7.13 * 10^6$	$5.63 * 10^7$	$1.58 * 10^4$			
1988	61	566 <small>384</small>	109	114	101	$1.21 * 10^7$	$6.04 * 10^7$	$2.69 * 10^4$			
1989	62	594 <small>416</small>	111	116	104	$2.06 * 10^7$	$6.47 * 10^7$	$4.58 * 10^4$			
1990	63	622 <small>448</small>	112	117	106	$3.51 * 10^7$	$6.93 * 10^7$	$7.80 * 10^4$			
1991	63	652 <small>448</small>	113	119	109	$5.97 * 10^7$	$7.43 * 10^7$	$1.33 * 10^5$			
1992	64	682 <small>480</small>	114	120	112	$1.02 * 10^8$	$7.96 * 10^7$	$2.26 * 10^5$			
1993	65	713 <small>512</small>	116	121	114	$1.73 * 10^8$	$8.54 * 10^7$	$3.84 * 10^5$			
1994	66	744 <small>544</small>	117	123	117	$2.94 * 10^8$	$9.15 * 10^7$	$6.53 * 10^5$			
1995	66	777 <small>544</small>	118	124	121	$5.00 * 10^8$	$9.81 * 10^7$	$1.11 * 10^6$			
1996	67	810 <small>576</small>	120	126	122	$8.51 * 10^8$	$1.05 * 10^8$	$1.89 * 10^6$			
1997	68	844 <small>608</small>	121	127	125	$1.45 * 10^9$	$1.13 * 10^8$	$3.22 * 10^6$			
1998	69	879 <small>640</small>	122	129	129	$2.46 * 10^9$	$1.21 * 10^8$	$5.48 * 10^6$			
1999	70	915 <small>672</small>	123	130	130	$4.19 * 10^9$	$1.29 * 10^8$	$9.31 * 10^6$			
2000	70	952 <small>704</small>	125	132	132	$7.13 * 10^9$	$1.39 * 10^8$	$1.58 * 10^7$			
2001	71	990 <small>736</small>	126	133	135	$1.21 * 10^{10}$	$1.49 * 10^8$	$2.70 * 10^7$			
2002	72	1028 <small>768</small>	127	135	139	$2.06 * 10^{10}$	$1.59 * 10^8$	$4.59 * 10^7$			
2003	73	1068 <small>800</small>	129	136	140	$3.51 * 10^{10}$	$1.71 * 10^8$	$7.80 * 10^7$			
2004	73	1108 <small>832</small>	130	138	143	$5.98 * 10^{10}$	$1.83 * 10^8$	$1.33 * 10^8$			
2005	74	1149 <small>864</small>	131	139	147	$1.02 * 10^{11}$	$1.96 * 10^8$	$2.26 * 10^8$			
2006	75	1191 <small>896</small>	133	141	148	$1.73 * 10^{11}$	$2.10 * 10^8$	$3.84 * 10^8$			

2007	76	1235 928	134	142	152	$2.94 * 10^{11}$	$2.25 * 10^8$	$6.54 * 10^8$
2008	76	1279 960	135	144	155	$5.01 * 10^{11}$	$2.41 * 10^8$	$1.11 * 10^9$
2009	77	1323 1024	137	145	157	$8.52 * 10^{11}$	$2.59 * 10^8$	$1.89 * 10^9$
2010	78	1369 1056	138	146	160	$1.45 * 10^{12}$	$2.77 * 10^8$	$3.22 * 10^9$
2011	79	1416 1088	139	148	163	$2.47 * 10^{12}$	$2.97 * 10^8$	$5.48 * 10^9$
2012	80	1464 1120	141	149	165	$4.19 * 10^{12}$	$3.19 * 10^8$	$9.32 * 10^9$
2013	80	1513 1184	142	151	168	$7.14 * 10^{12}$	$3.41 * 10^8$	$1.59 * 10^{10}$
2014	81	1562 1216	143	152	172	$1.21 * 10^{13}$	$3.66 * 10^8$	$2.70 * 10^{10}$
2015	82	1613 1248	145	154	173	$2.07 * 10^{13}$	$3.92 * 10^8$	$4.59 * 10^{10}$
2016	83	1664 1312	146	155	177	$3.51 * 10^{13}$	$4.20 * 10^8$	$7.81 * 10^{10}$
2017	83	1717 1344	147	157	180	$5.98 * 10^{13}$	$4.51 * 10^8$	$1.33 * 10^{11}$
2018	84	1771 1376	149	158	181	$1.02 * 10^{14}$	$4.83 * 10^8$	$2.26 * 10^{11}$
2019	85	1825 1440	150	160	185	$1.73 * 10^{14}$	$5.18 * 10^8$	$3.85 * 10^{11}$
2020	86	1881 1472	151	161	188	$2.94 * 10^{14}$	$5.55 * 10^8$	$6.54 * 10^{11}$
2021	86	1937 1536	153	163	190	$5.01 * 10^{14}$	$5.94 * 10^8$	$1.11 * 10^{12}$
2022	87	1995 1568	154	164	193	$8.52 * 10^{14}$	$6.37 * 10^8$	$1.89 * 10^{12}$
2023	88	2054 1632	156	166	197	$1.45 * 10^{15}$	$6.83 * 10^8$	$3.22 * 10^{12}$
2024	89	2113 1696	157	167	198	$2.47 * 10^{15}$	$7.32 * 10^8$	$5.48 * 10^{12}$
2025	89	2174 1728	158	169	202	$4.20 * 10^{15}$	$7.84 * 10^8$	$9.33 * 10^{12}$
2026	90	2236 1792	160	170	205	$7.14 * 10^{15}$	$8.41 * 10^8$	$1.59 * 10^{13}$
2027	91	2299 1856	161	172	207	$1.21 * 10^{16}$	$9.01 * 10^8$	$2.70 * 10^{13}$
2028	92	2362 1888	162	173	210	$2.07 * 10^{16}$	$9.66 * 10^8$	$4.59 * 10^{13}$
2029	93	2427 1952	164	175	213	$3.52 * 10^{16}$	$1.04 * 10^9$	$7.81 * 10^{13}$
2030	93	2493 2016	165	176	215	$5.98 * 10^{16}$	$1.11 * 10^9$	$1.33 * 10^{14}$
2031	94	2560 2080	167	178	218	$1.02 * 10^{17}$	$1.19 * 10^9$	$2.26 * 10^{14}$
2032	95	2629 2144	168	179	222	$1.73 * 10^{17}$	$1.27 * 10^9$	$3.85 * 10^{14}$
2033	96	2698 2208	169	181	223	$2.95 * 10^{17}$	$1.37 * 10^9$	$6.55 * 10^{14}$
2034	96	2768 2272	171	182	227	$5.01 * 10^{17}$	$1.46 * 10^9$	$1.11 * 10^{15}$
2035	97	2840 2336	172	184	230	$8.53 * 10^{17}$	$1.57 * 10^9$	$1.90 * 10^{15}$
2036	98	2912 2400	173	185	232	$1.45 * 10^{18}$	$1.68 * 10^9$	$3.22 * 10^{15}$
2037	99	2986 2464	175	186	235	$2.47 * 10^{18}$	$1.80 * 10^9$	$5.49 * 10^{15}$
2038	99	3061 2528	176	188	239	$4.20 * 10^{18}$	$1.93 * 10^9$	$9.33 * 10^{15}$
2039	100	3137 2592	178	189	240	$7.14 * 10^{18}$	$2.07 * 10^9$	$1.59 * 10^{16}$
2040	101	3214 2656	179	191	244	$1.22 * 10^{19}$	$2.22 * 10^9$	$2.70 * 10^{16}$
2041	102	3292 2720	180	192	247	$2.07 * 10^{19}$	$2.38 * 10^9$	$4.60 * 10^{16}$
2042	103	3371 2784	182	194	248	$3.52 * 10^{19}$	$2.55 * 10^9$	$7.82 * 10^{16}$
2043	103	3451 2880	183	195	252	$5.99 * 10^{19}$	$2.73 * 10^9$	$1.33 * 10^{17}$
2044	104	3533 2944	185	197	255	$1.02 * 10^{20}$	$2.93 * 10^9$	$2.26 * 10^{17}$
2045	105	3616 3008	186	198	257	$1.73 * 10^{20}$	$3.14 * 10^9$	$3.85 * 10^{17}$
2046	106	3700 3072	187	200	260	$2.95 * 10^{20}$	$3.36 * 10^9$	$6.55 * 10^{17}$
2047	106	3785 3168	189	201	264	$5.02 * 10^{20}$	$3.60 * 10^9$	$1.11 * 10^{18}$
2048	107	3871 3232	190	203	265	$8.53 * 10^{20}$	$3.86 * 10^9$	$1.90 * 10^{18}$
2049	108	3959 3328	192	204	269	$1.45 * 10^{21}$	$4.14 * 10^9$	$3.23 * 10^{18}$
2050	109	4047 3392	193	206	272	$2.47 * 10^{21}$	$4.44 * 10^9$	$5.49 * 10^{18}$

4.5. Equipment Cost Equivalent Key Sizes. Assuming the \$100 price for a stripped down PC (cf. (3.6)) and the resulting factor of 2500 are acceptable, Table 1 can be used to derive equipment cost equivalent key sizes in the following manner. A lower bound for the equipment cost for a successful one day attack is given in the second to last column of Table 1, in year y in dollars of y .

The symmetric key sizes are derived based on Hypothesis 1, and the EC key sizes are based on estimates that are cost consistent with the symmetric key sizes (cf. (3.6)), so for those systems no corrections are necessary.

For classical asymmetric systems, MY are supposedly 2500 times as expensive, which is, for our computational purposes only, equivalent to assuming that the DES offers acceptable security until about 1997, since $1997 - 1982 = 15$ and $2^{(15+23/30)}$ (cf. (4.1)) is close to 2500. Thus, using (4.4), classical asymmetric key sizes that are equipment cost equivalent to symmetric and EC key sizes for year y can be found in Table 1 in the classical asymmetric key size column for year $y - (23*15)/43 = y - 8$. The resulting key sizes, rounded up to the nearest multiple of 32, are given as the second entry in the classical asymmetric key sizes column of Table 1. Breaking such keys requires a substantially smaller number of MY than the infeasible number of MY for year y , but acquiring the required MY is supposed to be prohibitively expensive.

For subgroup discrete logarithm systems in year y , let t and s be the subgroup and finite field size, respectively, for year y , and let s' be the finite field size for year $y - 8$. For cost equivalence with symmetric and EC key sizes in year y use subgroups of size $t + 4*\log_2(s) - 4*\log_2(s')$ over finite fields of size s' . As a rule of thumb, subgroups of size $t + 2$ over finite fields of size s' will do. As an example, in the year 2000 the following key sizes are more or less equipment cost equivalent: 70-bit symmetric keys, 682-bit classical asymmetric keys, 127-bit subgroups with 682-bit finite fields, and 132-bit EC keys.

A similar straightforward analysis can be carried out for any other PC price one may prefer. For instance, for \$10 or \$1000 per PC the $y - 8$ should be changed into $y - 6$ or $y - 10$, respectively.

5 Practical Consequences of Table 1

5.1. DSS. DSS key sizes can be recommended for commercial applications only until the year 2002 for the field size, until 2013 for the hash function, and until 2026 for the subgroup size. For security until 2013, it is advisable to use the DSS with a 1513-bit finite field. Beyond 2013 the 160-bit size of SHA-1 may no longer be adequate. Changing it may force a change in the subgroup size that would otherwise not have been necessary until 2026. According to [16], NIST is working on a revision for the revision for the DSS, with key sizes as reported in Table 2 (and hash size matching the size of q).

Table 2
Proposed key sizes for the revised DSS

Size q	160	256	384	512
Size p	1024	3072	7680	15360

5.2. Effect on Cryptosystem Speed. RSA keys that are supposed to be secure until 2040 are about 3 times larger than popular 1024-bit RSA keys, making those large keys 9 to 27 times slower to use: 9 for signature verification or encryption with a fixed length public exponent, 27 for signature generation or decryption. TDL systems will slowdown by a factor 27 compared to 1024-bit versions. SDL systems slowdown

by about a factor 11 compared to currently secure SDL systems. The speed of EC systems is hardly affected. Within a few years faster processors will have solved these performance problems by our Moore assumption. Note, however, that this may not be the case in more restricted environments such as smartcards, where bandwidth and power consumption constraints also have a more limiting effect on key sizes.

5.3. 512-bit RSA Keys. RSA keys of 512 bits are widely used all over the Web, e.g. in SSL protected communications. According to Table 1, such keys should not have been used beyond 1986. A 512-bit RSA key was factored in August 1999 (cf. [5]).

5.4. 768-bit RSA Keys. According to Table 1 usage of 768-bit RSA keys can no longer be recommended. Even in the equipment cost equivalent model 768-bit RSA keys will soon no longer offer security comparable to the security of the DES in 1982.

5.5. RSA and EC. It is often informally argued that 1024-bit RSA and 160-bit EC systems offer more or less the same level of security. This is not what one would conclude from Table 1. We discuss several aspects of this contentious issue in [15].

5.6. SDL and EC. The gap between the suggested SDL and EC key sizes widens slowly due to the rapidly growing size of the finite fields in SDL.

5.7. Effectiveness of Guessing. The sizes suggested in Table 1 for the year 2000 or later are in practice infeasible to guess.

5.8. Effectiveness of Incomplete Attacks. As shown in [15], incomplete attacks can on average not be expected to pay off.

5.9. Effectiveness of Elliptic Curve Method. As shown in [15], the Elliptic Curve Method cannot be expected to break keys of the suggested sizes.

5.10. Wassenaar Arrangement for Mass Market Applications. The WA allows 64-bit symmetric keys and 512-bit classical asymmetric keys for mass market applications. According to Table 1 it is advisable to increase the 512-bit bound for classical asymmetric keys to 672 or 768 bits.

Acknowledgements. The authors want to thank Joe Buhler, Bruce Dodson, Stuart Haber, Paul Leyland, Alfred Menezes, Andrew Odlyzko, Michael Wiener, and Paul Zimmermann for their helpful remarks.

References

1. Eli Biham, A fast new DES implementation in software.
2. M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener, Minimal key lengths for symmetric ciphers to provide adequate commercial security, www.bsa.org/policy/encryption/cryptographers_c.html, January 1996.
3. A. Bosselaers, Even faster hashing on the Pentium, manuscript, Katholieke Universiteit Leuven, May 13, 1997.
4. J.R.T. Brazier, Possible NSA decryption capabilities, <http://jya.com/nsa-study.htm>.

5. S. Cavallar, B. Dodson, A.K. Lenstra, B. Murphy, P.L. Montgomery, H.J.J. te Riele, et al., Factorization of a 512-bit RSA modulus, manuscript, October 1999.
6. www.counterpane.com/speed.html.
7. M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, J.J. Quisquater, Efficient hardware and software implementations of the DES, Proceedings Crypto'84.
8. W. Diffie, BNR Inc. report, 1980.
9. W. Diffie, E. Hellman, Exhaustive cryptanalysis of the NBS Data Encryption Standard, Computer, v. 10 (1977), 74-84.
10. B. Dixon, A.K. Lenstra, Factoring integers using SIMD sieves, Proceedings Eurocrypt'93, LNCS 765, 28-39.
11. Electronic Frontier Foundation, Cracking DES, O'Reilly, July 1998.
12. Rob Gallant, personal communication, August 1999.
13. P.C. Kocher, Breaking DES, RSA Laboratories' Cryptobytes, v. 5, no 2 (1999); also at www.rsa.com/rsalabs/pubs/cryptobytes.
14. P.C. Kocher, personal communication, September 1999.
15. A.K. Lenstra, E.R. Verheul, Selecting Cryptographic Key Sizes, submitted for publication, September 1999; available at www.cryptosavvy.nl.
16. A.J. Menezes, personal communication, September 1999.
17. P.L. Montgomery, letter to the editor of IEEE Computer, August 1999.
18. V.I. Nechaev, Complexity of a determinate algorithm for the discrete logarithm, Mathematical Notes, 55 (2) 1994, 155-172. Translated from Matematicheskie Zametki, 55(2), 91-101, 1994. This result dates back from 1968.
19. Tiniest circuits hold prospect of explosive computer speeds, The New York Times, July 16, 1999; Chip designers look for life after silicon, The New York Times, July 19, 1999.
20. A.M. Odlyzko, The future of integer factorization, RSA Laboratories' Cryptobytes, v. 1, no. 2 (1995), 5-12; also at www.research.att.com/~amo/doc/crypto.html or www.rsa.com/rsalabs/pubs/cryptobytes.
21. P.W. Shor, Algorithms for quantum computing: discrete logarithms and factoring, Proceedings of the IEEE 35th Annual Symposium on Foundations of Computer Science, 124-134, 1994.
22. V. Shoup, Lower bounds for discrete logarithms and related problems, Proceedings Eurocrypt'97, LNCS 1233, 256-266.
23. R.D. Silverman, rump session presentation at Crypto'97.
24. R.D. Silverman. Exposing the Mythical MIPS Year, IEEE Computer, August 1999, 22-26.
25. P.C. van Oorschot, M.J. Wiener, Parallel collision search with cryptanalytic applications, Journal of Cryptology, v. 12 (1999), 1-28.
26. M.J. Wiener, Efficient DES key search, manuscript, Bell-Northern Research, August 20, 1993.
27. M.J. Wiener, Performance Comparison of Public-Key Cryptosystems, RSA Laboratories' Cryptobytes, v. 4, no. 1 (1998), 1-5; also at www.rsa.com/rsalabs/pubs/cryptobytes.
28. M.J. Wiener, personal communication, 1999.

A Structured ElGamal-Type Multisignature Scheme

Mike Burmester¹, Yvo Desmedt², Hiroshi Dohi³, Masahiro Mambo⁴,
Eiji Okamoto⁵, Mitsuru Tada⁶, and Yuko Yoshifujii^{6,*}

¹ Information Security Group, Royal Holloway, University of London
m.burmester@rhbnc.ac.uk

² Department of Computer Science, Florida State University
desmedt@cs.fsu.edu

³ Department of Mathematics, Faculty of Science, Okayama University
hdoi@math.okayama-u.ac.jp

⁴ Education Center for Information Processing & Graduate School of Information Sciences, Tohoku University, mambo@ecip.tohoku.ac.jp

⁵ Center for Cryptography, Computer and Network Security
University of Wisconsin, Milwaukee, okamoto@cs.uwm.edu

⁶ School of Information Science, Japan Advanced Institute of Science and Technology,
{mt,yosifujii}@jaist.ac.jp

Abstract. We propose a structured multisignature scheme which is based on a modified ElGamal signature scheme and analyze its security. The structure takes into account the order of the signers. With serial structures, different signing orders produce different multisignatures. In contrast, with parallel structures the multisignatures are independent of the signing order. Our structured multisignatures can deal with structures which are composed of serial and parallel signing orders. We give reductions for the security of the proposed scheme, and for the specified order of the signers in the serial and mixed cases.

Keywords: Multisignature, Structured multisignature, Group structure, Series-parallel graph, ElGamal signature.

1 Introduction

When multiple entities sign a document, the signing order often reflects the role of each signer and signatures with different signing orders are regarded as multisignatures with different meanings. A typical example is a multisignature in a company. Usually, the head of a section should sign a document after the other members of the section have signed it. Other examples involve banks and command structures.

Of course the signing order is of little relevance to authentication. However there are other aspects one should take into account, such as the liability of the

*Present affiliation of the last author: NTT Service Integration Laboratories

signers. This could be related to their ranking and determined by the signing order. Depending on the application, a different signing order may be required. Moreover, each signer may only wish to sign after the previous signers have done so, and the verifier may require that the correct order has been adhered to.

We can consider two cases for the signing order: 1) *serial signing*, in which the signing order can be detected by a verifier from a signature, 2) *parallel signing*, in which the signing order cannot be detected by a verifier from a signature. A multisignature scheme is said to be *structured* if the group of signers is structured. The structure takes into account the signing order of the entities of the signing groups, if this is serial. In this case different signing orders yield different multisignatures. In contrast, the multisignature for parallel signing orders is group independent. Our structured multisignature scheme can deal with group structures which are composed of serial and parallel signing orders.

In this paper we propose a structured multisignature scheme based on a modified ElGamal signature scheme and give reductions for its security, including the security for the specified order of the signers. This paper is organized as follows. After this introduction we discuss related work in Section 2. In Section 3 we represent signer groups by series-parallel graphs and in Section 4 we give definitions and specify our setting. In Section 5 we define our model, and the reductions and functions which we shall use for the security proof. After describing the basic modified ElGamal signature scheme in Section 6, we present a serial multisignature scheme in Section 7, a parallel multisignature scheme in Section 8, and a mixed multisignature scheme in Section 9. In Section 10 we modify our scheme to get a more efficient scheme by exploiting the decomposition tree of series-parallel graphs. Finally, conclusions are given in Section 11.

2 Related Work

So far there are several proposals for multisignature schemes, as for example in [Oka88, OO91, HZ92, Shim94, DOMU94, HMP95, DOM98, OO99]. Multisignature schemes have advantages over multiple iterations of a single signature scheme either in the length of the signature or in the computational cost of generating and verifying the signature. Another property one should pay attention to is the signing order. Some multisignature schemes are independent of the order of the signers [OO91, HZ92, Shim94, HMP95], while other schemes are order specified [Oka88, DOMU94, DOM98, OO99].

Among the schemes which are sensitive to the signing order, the scheme in [Oka88] is constructed using bijective functions such as the RSA signing function [RSA78]. The scheme in [DOMU94] is also based on the RSA signature scheme, but this scheme has been shown to be insecure. A modified version of this scheme is proposed in [DOM98], however its security analysis is not complete. Another scheme [DOM98] uses an ElGamal-type signature [EIG85, HMP94], but as with the RSA based scheme, its security analysis is not complete.

The Ohta-Okamoto schemes in [OO91] are converted from corresponding identification schemes such as the Schnorr scheme [Schn91] and the Fiat-Shamir

scheme [FS86]. For single signatures, the security proof for the Schnorr signature scheme and a modified ElGamal signature scheme are given in [PS96] using the Random Oracle model [BR93] (however there are some problems with this model [CGH98]). Ohta-Okamoto prove the security of their schemes using an ID reduction to an identification scheme [OO91]. For single signatures their proof gives a tighter reduction than [PS96]. However the Ohta-Okamoto schemes do not consider mixed structures composed of serial and parallel signing orders. In particular, the security of multisignature schemes for such a composite group structure has not been proven.

A structured multisignature scheme based on an ElGamal-type signature is also proposed in [DOM98] but its security for the signing order is not complete.

3 Series-Parallel Graph

We represent the group of signers by a graph. We consider directed graphs which satisfy the following conditions:

- (G-1) Each signer corresponding to an edge appears only once in the graph.
- (G-2) The graphs are restricted to series-parallel type graphs.

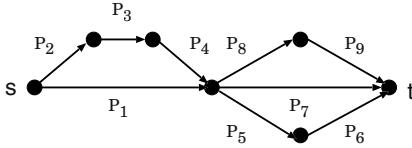
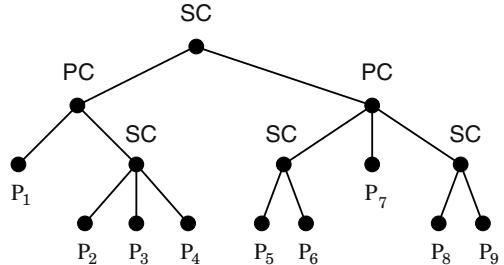
A *series-parallel graph* is a graph which is generated by series and parallel compositions of series-parallel graphs. The simplest series-parallel graph is a base graph of two vertices and an edge. We refer the reader to [Len90] for more details. The set of all graphs satisfying the above conditions is denoted by *SPG*.

In Figure 1 we illustrate a typical series-parallel graph. A series-parallel graph can be decomposed into a unique tree called the decomposition tree of the graph. In Figure 2 we show the decomposition tree of the graph in Figure 1. The labels of the vertices show the type of composition of the subordinate edges. Vertices labeled *SC* show series compositions whereas vertices labeled *PC* show parallel compositions. Series-parallel graphs can be recognized in linear time and the decomposition tree of a series-parallel graph can be found in linear time [Len90].

4 Notation and Definition

We will use a discrete logarithm setting. Let p and q be appropriate primes such that $q|(p - 1)$, and let $g \in \mathbb{Z}_p^*$ be a q -th root of 1 modulo p . P_i is a signer with secret key $a_i \in \mathbb{Z}_q^*$, and k_i is a random number in \mathbb{Z}_q^* selected by P_i . $M \in \{0, 1\}^*$ is the message to be signed. $a \in_R A$ indicates that the element a is selected from the set A uniformly at random.

Constructible groups. Let $\mathcal{G} \subset SPG$ be a collection of signer groups, $G_k \in \mathcal{G}$ a signer group represented by a series-parallel graph, P_i a signer in G_k , $a_i \in \mathbb{Z}_q^*$ the secret key of P_i , and $y_i^{(k)} \in \mathbb{Z}_p^* \setminus \{1\}$ the partial public key of P_i corresponding to a_i . We say that \mathcal{G} is an *available group* of a signature

**Fig. 1.** A series-parallel graph G **Fig. 2.** The decomposition tree of G

scheme if there is a set of secret keys $\{a_i | P_i \in G_k, G_k \in \mathcal{G}\}$ of this scheme such that,

$$y_i^{(k)} \neq y_j^{(l)} \quad \text{for } \forall G_k, G_l \in \mathcal{G}, \forall (P_i, P_j) \in (G_k, G_l), i \neq j. \quad (1)$$

We allow $k = l$ in (1). The set of keys $\{a_i | P_i \in G_k, G_k \in \mathcal{G}\}$ of an available group is called an *appropriate secret-key set* for \mathcal{G} or simply, *appropriate for* \mathcal{G} . If it is feasible to generate an appropriate secret-key set from an available group \mathcal{G} , then \mathcal{G} is called a *constructible group*.

Signature structure. When signers sign a document sequentially, the signing order has a special meaning. We call such a signature structure, *serial*. On the other hand, when all or a part of the signers create a partial signature for a document in arbitrary order and a complete signature is created from these partial signatures, the signing order of the signers has no meaning. We call such a signature structure, *parallel*.

The signer group of a serial structure is denoted by (P_1, P_2, \dots, P_n) . This means that P_1 signs first, P_2 second and so on. After all P_1, P_2, \dots, P_{n-1} have signed, P_n signs. For simplicity, we write

$$\begin{aligned} G_{i, \dots, j} &= (\dots, P_{i-1}, P_i, \dots, P_{j-1}, P_j, \dots), \\ G_{j, \dots, i} &= (\dots, P_{i-1}, P_j, \dots, P_{j-1}, P_i, \dots), \end{aligned}$$

where $1 \leq i < j \leq n$.

The signer group of a parallel structure is denoted by $\wedge(P_1, P_2, \dots, P_n)$.

In the scheme we propose the partial signature created by signer P_i is a triple $(s_i, r_i, r) \in Z_q \times Z_p^* \times Z_p^*$. The partial signature created by the last signer P_n is the multisignature of the entire group. For (s_n, r_n, r) we have $r_n = r$, so that the final form of the multisignature is (s_n, r_n) .

Let $i, j \in \{1, 2, \dots, n\}$, $i < j$, and $K \subset \{1, 2, \dots, n\}$. For a variable v , let $v_{[i,j,K]}$ denote the sequence v_i, v_{i+1}, \dots, v_j in which all v_l , $l \in K$, are excluded. $v_{[1,n,\phi]}$ is simply denoted by $v_{[i,j]}$.

For simplicity, $(\text{mod}p)$ is sometimes omitted in this paper.

Hash function. In this paper we shall use a *target-collision intractable* hash function, hash , which takes values of arbitrary length as input and outputs a value in \mathbf{Z}_q^* . (A hash function is target-collision intractable if for a given x it is infeasible to compute a y such that $\text{hash}(x) = \text{hash}(y)$.)

5 Attacks and Reductions

Attack model. For our security analysis, we consider a model in which all insiders (signers) except an honest signer may collude in the attack. The attackers give a partial signature of a message M to the honest signer P_i and obtain a valid partial signature by P_i of M . With this information, the attackers try to obtain forged signatures or to derive P_i 's secret key.

Reductions. We use the following reductions in our security proof (*cf.* [Woll87]):

- $f \leq_m^p g$ denotes that f reduces to g with respect to the polynomial-time many-one (\leq_m^p -) reducibility. (That is, there exists a polynomial-time computable function h such that $f(x) = g(h(x))$.)
- $f \leq_{k-tt}^p g$ denotes that f reduces to g with respect to the polynomial-time truth-table (\leq_{k-tt}^p -) reducibility. In a \leq_{k-tt}^p -reducibility only k non-adaptive queries to an oracle are allowed. If we do not wish to stress the number of queries, we simply write $f \leq_{tt}^p g$.
- $f \leq_T^p g$ denotes that f reduces to g with respect to the polynomial-time Turing (\leq_T^p -) reducibility. In a \leq_T^p -reducibility, a polynomial-time bounded oracle Turing machine can access an oracle adaptively.

For $\text{type} \in \{m, k-tt, T\}$, if $f \leq_{\text{type}}^p g$ and $g \leq_{\text{type}}^p f$, then $f \equiv_{\text{type}}^p g$.

We use the following functions for the proof of security.

- **DLP_q**(X, g, p, q) is a function that on input two primes p, q with $q|(p - 1)$, $X \in \mathbf{Z}_p^*$, $g \in \mathbf{Z}_p^*$ of order q , outputs $x \in \mathbf{Z}_q$ such that $X \equiv g^x(\text{mod}p)$, if such an x exists. This function solves the discrete logarithm problem in a subgroup of prime order q .
- **Forge**(y, r, M, g, p, q) is a function that on input two primes p, q with $q|(p - 1)$, $y \in \mathbf{Z}_p^*$, $r \in \mathbf{Z}_p^*$ satisfying $\text{gcd}(r, q) = 1$, $M \in \{0, 1\}^*$, $g \in \mathbf{Z}_p^*$ of order q , outputs $s' \in \mathbf{Z}_q$ with $g^{s'} \equiv yr^{r \cdot \text{hash}(r, M)}(\text{mod}p)$ for a given hash function $\text{hash}()$, if such an s' exists.
This function forges a valid signature s' using the pair (M, r) of a message M and a number r and available public information, but does not use the signer's secret key.
- **Secret1**(y, r, s, M, g, p, q) is a function that on input two primes p, q with $q|(p - 1)$, $y \in \mathbf{Z}_p^*$, $r \in \mathbf{Z}_p^*$ with $\text{gcd}(r, q) = 1$, $s \in \mathbf{Z}_q$, $M \in \{0, 1\}^*$, $g \in \mathbf{Z}_p^*$ of order q , outputs $(a, k) \in (\mathbf{Z}_q^*, \mathbf{Z}_q^*)$ such that

$$y \equiv g^a, r \equiv g^k, s \equiv a + kr \cdot \text{hash}(r, M) \pmod{q}$$

for a given hash function $\text{hash}()$, if such a pair of (a, k) exists.

This function computes the secret key a of P and the random number k using a valid signature (r, s) by P and available public information.

- **Secret2** $[(P_1, P_2, \dots, P_n)](y_i, r_i, \tilde{r}, s_i, \tilde{k}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q)$ is a function that on input two primes p, q with $q|(p-1)$, $y_i, r_i \in \mathbb{Z}_p^*$, $\tilde{r} \in \mathbb{Z}_p^*$ satisfying $\text{gcd}(\tilde{r}, q) = 1$, $s_i, \tilde{k} \in \mathbb{Z}_q$, $a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]} \in \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*$, $M \in \{0, 1\}^*$, $g \in \mathbb{Z}_p^*$ of order q , outputs $(a_i, k_i) \in (\mathbb{Z}_q^*, \mathbb{Z}_q^*)$ such that $a_{[1,n]}$ is appropriate for (P_1, P_2, \dots, P_n) , and

$$\alpha = (\dots((a_1 + 1)a_2 + 1)\dots)a_{i-1} \pmod{q},$$

$$y_i \equiv g^{(\alpha+1)a_i}, r_i \equiv (g^{\tilde{k}})^{a_i} \cdot g^{k_i},$$

$$s_i \equiv ((\alpha + \tilde{k}\tilde{r} \cdot \text{hash}(\tilde{r}, M)) + 1)a_i + k_i\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q},$$

for a given hash function $\text{hash}()$, if such a pair of (a_i, k_i) exists, and otherwise outputs \perp .

This function computes the pair (a_i, k_i) consisting of the secret key a_i and the random number k_i of an honest signer P_i in the following situation. All signers except P_i collude to compute (a_i, k_i) , using any available public information and a valid partial signature (r_i, s_i, r) created by P_i in a serial structure for a message M . The colluding entities do not need to follow the signature creation procedure for computing these values. A partial signature $(\tilde{r}_{i-1}, \tilde{s}_{i-1}, \tilde{r})$ given to P_i satisfies $y_{i-1}\tilde{r}_{i-1}^{\text{hash}(\tilde{r}, M)} \equiv g^\alpha \cdot g^{\tilde{k}\tilde{r} \cdot \text{hash}(\tilde{r}, M)} \equiv g^{\tilde{s}_{i-1}}$, where $\tilde{r}_{i-1} \equiv g^{\tilde{k}}, \tilde{s}_{i-1} \equiv \alpha + \tilde{k}\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q}$.

- **Flip** $[G_{i,\dots,j}, G_{j,\dots,i}](y_{i-1}, y_i, y', \tilde{r}_{i-1}, r_i, \tilde{r}, \tilde{s}_{i-1}, s_i, \tilde{k}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q)$ is a function that on input two primes p, q with $q|(p-1)$, $y_{i-1}, y_i, y', \tilde{r}_{i-1}, r_i \in \mathbb{Z}_p^*$, $\tilde{r} \in \mathbb{Z}_p^*$ satisfying $\text{gcd}(\tilde{r}, q) = 1$, $\tilde{s}_{i-1}, s_i, \tilde{k} \in \mathbb{Z}_q$, $a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]} \in \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*$, $M \in \{0, 1\}^*$, $g \in \mathbb{Z}_p^*$ of order q , outputs $(s', r') \in (\mathbb{Z}_q, \mathbb{Z}_p^*)$ such that $s' \in \mathbb{Z}_q, r' \in \mathbb{Z}_p^*$ with $\text{gcd}(r', q) = 1$, $\alpha' \in \mathbb{Z}_q^*$, $a_i, k_i \in \mathbb{Z}_q^*, a_{[1,n,\{i\}]}$ is appropriate for $[G_{i,\dots,j}, G_{j,\dots,i}]$, and

$$\alpha = (\dots((a_1 + 1)a_2 + 1)\dots)a_{i-1} \pmod{q},$$

$$\beta = \alpha a_j a_{i+1} \dots a_{j-1} a_{j+1} \dots a_n + a_j a_{i+1} \dots a_{j-1} a_{j+1} \dots a_n$$

$$+ a_{i+1} \dots a_{j-1} a_{j+1} \dots a_n + \dots$$

$$+ a_{j-1} a_{j+1} \dots a_n + a_{j+1} \dots a_n \pmod{q},$$

$$\gamma = a_{j+1} \dots a_n + \dots + a_n \pmod{q},$$

$$y_{i-1} \equiv g^\alpha, y_i \equiv (y_{i-1} \cdot g)^{a_i}, y' \equiv g^{\beta a_i + \gamma},$$

$$\tilde{r}_{i-1} \equiv g^{\tilde{k}}, r_i \equiv \tilde{r}_{i-1}^{a_i} \cdot g^{k_i}, r' \equiv g^{\alpha'},$$

$$\tilde{s}_{i-1} \equiv \alpha + \tilde{k}\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q},$$

$$s_i \equiv (\tilde{s}_{i-1} + 1)a_i + k_i\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q},$$

$$s' \equiv (\beta a_i + \gamma) + \alpha' r' \cdot \text{hash}(r', M) \pmod{q}$$

for a given hash function $\text{hash}()$, if such a pair of (s', r') exists.

This function forges a signature of a signer structure $G_{j,\dots,i}$ from the signature of a signer structure $G_{i,\dots,j}$ in the following situation. All signers except the honest signer P_i collude in the forgery, using public information and a valid partial signature (r_i, s_i, r) created by P_i for the message M . The colluding entities do not need to follow the signature creation procedure for

computing these values. A partial signature $(\tilde{r}_{i-1}, \tilde{s}_{i-1}, \tilde{r})$ given to P_i satisfies $y_{i-1}\tilde{r}_{i-1}^{\text{hash}(\tilde{r}, M)} \equiv g^\alpha \cdot g^{k\tilde{r} \cdot \text{hash}(\tilde{r}, M)} \equiv g^{\tilde{s}_{i-1}}$, where $\tilde{r}_{i-1} \equiv g^{\tilde{k}}, \tilde{s}_{i-1} \equiv \alpha + \tilde{k}\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q}$.

- **Forge** $[G_o, G_f](y_{i-1}, y_i, y', \tilde{r}_{i-1}, r_i, \tilde{r}, \tilde{s}_{i-1}, s_i, \tilde{\beta}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q)$ is a function that on input two primes p, q with $q|(p-1)$, $y_{i-1}, y_i, y' \in \mathbb{Z}_p^*$, $\tilde{r}_{i-1} \in \mathbb{Z}_p^*$ satisfying $\text{gcd}(\tilde{r}_{i-1}, q) = 1$, $r_i \in \mathbb{Z}_p^*$ satisfying $\text{gcd}(r_i, q) = 1$, $\tilde{s}_{i-1}, s_i \in \mathbb{Z}_q$, $\tilde{\beta} \in \mathbb{Z}_q^*$, $a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]} \in \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*$, $M \in \{0, 1\}^*$, $g \in \mathbb{Z}_p^*$ of order q , outputs $(s', r') \in (\mathbb{Z}_q, \mathbb{Z}_p^*)$ such that $s' \in \mathbb{Z}_q$, $r' \in \mathbb{Z}_p^*$ satisfying $\text{gcd}(r', q) = 1$, $\beta' \in \mathbb{Z}_q$, $a_i, k_i \in \mathbb{Z}_q^*$, $a_{[1,n]}$ is appropriate for $\mathcal{G} \subset SPG$, function f_α is determined from $G_o \in \mathcal{G}$, $f_{\alpha'_1}$ and $f_{\alpha'_2}$ are determined from $G_f \in \mathcal{G}$, $P_i \in G_o$, $P_i \in G_f$, and

$$\begin{aligned} y_{i-1} &\equiv g^\alpha, \quad y_i \equiv (y_{i-1} \cdot g)^{a_i}, \quad y' \equiv g^{\alpha'_1 a_i + \alpha'_2}, \\ \alpha &= f_\alpha(a_{[1,n,\{i\}]}), \quad \alpha'_1 = f_{\alpha'_1}(a_{[1,n,\{i\}]}), \quad \alpha'_2 = f_{\alpha'_2}(a_{[1,n,\{i\}]}), \\ \tilde{r}_{i-1} &\equiv g^{\tilde{\beta}}, \quad r_i \equiv \tilde{r}_{i-1}^{a_i} \cdot g^{k_i}, \quad r' \equiv g^{\beta'}, \\ \tilde{s}_{i-1} &\equiv \alpha + \tilde{\beta}\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q}, \\ s_i &\equiv (\tilde{s}_{i-1} + 1)a_i + k_i \tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q}, \\ s' &\equiv \alpha'_1 a_i + \alpha'_2 r' \cdot \text{hash}(r', M) \pmod{q}, \end{aligned}$$

for a given hash function $\text{hash}()$, if such a pair of (s', r') exists.

This function forges the signature of a signer structure G_f from the signature of a signer structure G_o in the following situation. All signers except the honest signer P_i collude in the forgery using public information and a valid partial signature (r_i, s_i, r) created by P_i for the message M . The colluding entities do not need to follow the signature creation procedure for computing these values. A partial signature $(\tilde{r}_{i-1}, \tilde{s}_{i-1}, \tilde{r})$ given to P_i satisfies $y_{i-1}\tilde{r}_{i-1}^{\text{hash}(\tilde{r}, M)} \equiv g^\alpha \cdot g^{\tilde{\beta}\tilde{r} \cdot \text{hash}(\tilde{r}, M)} \equiv g^{\tilde{s}_{i-1}}$, where $\tilde{r}_{i-1} \equiv g^{\tilde{\beta}}, \tilde{s}_{i-1} \equiv \alpha + \tilde{\beta}\tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q}$.

- $FC(v, w)$ is a function which on input v, w , outputs the first component v .

6 Modified ElGamal Signature

We use a modified version [YL93] of the ElGamal signature scheme [EIG85]. To explain this scheme we consider a single entity signer group consisting of only P . The message is M . Let $a \in \mathbb{Z}_q^*$ be the secret key and $y = g^a \pmod{p}$ the public key of P .

Basic signature protocol

1. P selects $k \in_R \mathbb{Z}_q^*$ and computes $r = g^k \pmod{p}$. P repeats this process until r satisfies $\text{gcd}(r, q) = 1$.
2. P computes $s = a + kr \cdot \text{hash}(r, M) \pmod{q}$. The signature of M is (s, r) .
3. The signature (s, r) is verified by checking that $g^s \stackrel{?}{\equiv} yr^{r \cdot \text{hash}(r, M)} \pmod{p}$.

A valid signature passes the verification check because

$$g^s \equiv g^a \cdot g^{kr \cdot \text{hash}(r, M)} \equiv yr^{r \cdot \text{hash}(r, M)} \pmod{p}.$$

6.1 Security Considerations

Generation of a forged signature (r, s) : To show the difficulty of forging a signature we first consider the case when r is given.

- **Forge** $\leq_m^p \mathbf{DLP}_q$: $\mathbf{Forge}(y, r, M, g, p, q) = \mathbf{DLP}_q(yr^{r \cdot \text{hash}(r, M)}, g, p, q)$
- **DLP_q** \leq_m^p **Forge** : $\mathbf{DLP}_q(g^s, g, p, q) = \mathbf{Forge}(y, r, M, g, p, q)$, where $r \in_R Z_p^*$, $\gcd(r, q) = 1$, $M \in_R \{0, 1\}^*$, $y = g^s \cdot r^{-r \cdot \text{hash}(r, M)}$.

Observe that

$$y \cdot r^{r \cdot \text{hash}(r, M)} \equiv (g^s \cdot r^{-r \cdot \text{hash}(r, M)}) \cdot r^{r \cdot \text{hash}(r, M)} \equiv g^s.$$

Thus $\mathbf{DLP}_q \equiv_m^p \mathbf{Forge}$.

This proof shows only one aspect of the difficulty of signature forgery.

Security of secret values

- **Secret1** $\leq_{2-tt}^p \mathbf{DLP}_q$:
 $\mathbf{Secret1}(y, r, s, M, g, p, q) = (\mathbf{DLP}_q(y, g, p, q), \mathbf{DLP}_q(r, g, p, q))$
- **DLP_q** \leq_{1-tt}^p **Secret1** : $\mathbf{DLP}_q(g^a, g, p, q) = (r \cdot \text{hash}(r, M))^{-1} \cdot (-s(1 - r \cdot \text{hash}(r, M)) + FC(\mathbf{Secret1}(y, r, s, M, g, p, q))) \pmod{q}$, where $s \in_R Z_q^*$, $M \in_R \{0, 1\}^*$, $r = (g^a)^{-1} \cdot g^s$, $\gcd(r \cdot \text{hash}(r, M), q) = 1$, and $y = g^s \cdot r^{-r \cdot \text{hash}(r, M)}$.

Observe that

$$y \cdot r^{r \cdot \text{hash}(r, M)} = g^s,$$

and that

$$\begin{aligned} y &\equiv g^s \cdot r^{-r \cdot \text{hash}(r, M)} \equiv g^s \cdot g^{(-a+s)(-r\text{hash}(r, M))} \\ &\equiv g^{s(1-r\text{hash}(r, M))+ar\cdot\text{hash}(r, M)}. \end{aligned}$$

Thus $\mathbf{DLP}_q \equiv_{tt}^p \mathbf{Secret1}$.

7 Serial Multisignature Scheme

A partial public key y_i of P_i of a multisignature scheme must belong to $Z_p^* \setminus \{1\}$ and must also satisfy the condition (1). This is achieved by selecting appropriate secret-key groups.

For example, an appropriate secret-key group (a_1, \dots, a_n) for (P_1, \dots, P_n) is obtained by choosing $(\alpha_{i-1} + 1)a_i \neq a_j$ for $i = 2, \dots, n$, $j = 1, \dots, i-1$, where $\alpha_{i-1} = (\dots((a_1 + 1)a_2 + 1) \dots) a_{i-1} \pmod{q}$ (See Section 7.2).

Suppose n signers P_i sign a message M sequentially. Let $a_i \in Z_q^*$ be the secret key of P_i . Then the partial public key y_i of P_i in (P_1, \dots, P_n) , $i = 1, 2, \dots, n$, is computed as follows:

$$y_1 = g^{a_1}, \quad y_i = (y_{i-1} \cdot g)^{a_i}.$$

The public key of the group (P_1, \dots, P_n) is $y = y_n \pmod{p}$. The secret and public keys are generated either by a trusted center or by each signer using distributed

protocols. In the latter case, each signer repeatedly selects his secret key until the keys form an appropriate set. Furthermore, each signer needs to prove that they know the secret key which corresponds to the partial public key computed. For this purpose they may use the protocol in [Schn91], which does not reveal any secret information.

Serial multisignature protocol

1. **Generation of r .** P_1, \dots, P_n generate r together as follows.

- (a) P_1 selects $k_1 \in_R \mathbb{Z}_q^*$ and computes $r_1 = g^{k_1} \pmod{p}$. P_1 repeats this process until r_1 satisfies $\gcd(r_1, q) = 1$.
- (b) For $i \in \{2, \dots, n\}$:
 - P_{i-1} gives r_{i-1} to P_i .
 - P_i selects $k_i \in_R \mathbb{Z}_q^*$ and computes $r_i = r_{i-1}^{a_i} g^{k_i} \pmod{p}$; P_i repeats this process until r_i satisfies $\gcd(r_i, q) = 1$.
 - (c) $r = r_n$.

2. **Generation of s .** P_1, \dots, P_n generate s together as follows.

- (a) P_1 computes $s_1 = a_1 + k_1 r \cdot \text{hash}(r, M) \pmod{q}$.
 - (b) For $i \in \{2, \dots, n\}$:
 - P_{i-1} gives s_{i-1} to P_i .
 - P_i verifies that $g^{s_{i-1}} \equiv y_{i-1} r_{i-1}^{r \cdot \text{hash}(r, M)} \pmod{p}$ and if so computes
- $$s_i = (s_{i-1} + 1)a_i + k_i r \cdot \text{hash}(r, M) \pmod{q}.$$

- (c) $s = s_n$.

Multisignature. (r, s) is the multisignature on M by (P_1, \dots, P_n) .

Verification. A multisignature (r, s) is verified by checking the congruence

$$g^s \stackrel{?}{=} yr^{r \cdot \text{hash}(r, M)} \pmod{p}.$$

7.1 Security Considerations

Security of a_i and k_i . We consider the case when all the signers P_j except an honest signer P_i , $i > 1$, collude to derive a_i and k_i from any information they can obtain.

Let α be the exponent of y_{i-1} to the base g modulo p , i.e. $y_{i-1} = g^\alpha$, $\alpha = (\dots((a_1 + 1)a_2 + 1)\dots)a_{i-1} \pmod{q}$. Likewise, let \tilde{k} be the exponent of \tilde{r}_{i-1} to the base g modulo p , i.e. $\tilde{r}_{i-1} = g^{\tilde{k}}$.

- **Secret2** $[(P_1, \dots, P_i, \dots, P_n)] \leq_T^p \mathbf{DLP}_q$:
- Secret2** $[(P_1, \dots, P_i, \dots, P_n)](y_i, r_i, \tilde{r}, s_i, \tilde{k}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q) = (A, \mathbf{DLP}_q(r_i \cdot (g^{-A\tilde{k}}), g, p, q))$, where $A = \mathbf{DLP}_q(y_i, g^{\alpha+1}, p, q)$.

- $\mathbf{DLP}_q \leq_m^p \mathbf{Secret2}[(P_1, \dots, P_i, \dots, P_n)]$:

At first $a_{[1,i-1]} \in \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*$ is selected so that it is appropriate for $(P_1, \dots, P_i, \dots, P_{i-1})$. Then $a_{[i+1,n]} \in_R \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*$ is randomly selected. This procedure is repeated until the set of keys $a_{[1,n,\{i\}]}$ and a_i becomes appropriate for $(P_1, \dots, P_i, \dots, P_n)$. We can recognize that $a_{[1,n]}$ is not appropriate from the output of $\mathbf{Secret2}[(P_1, \dots, P_i, \dots, P_n)]$. In Section 7.2 we shall see that this procedure is feasible (with probability almost one) if n is bounded by a polynomial in $|p|$. When $a_{[1,n]}$ becomes appropriate, $\mathbf{DLP}_q(g^{a_i}, g, p, q)$ is computed in the following way:

$$\begin{aligned} \mathbf{DLP}_q(g^{a_i}, g, p, q) &= FC(\mathbf{Secret2}[(P_1, \dots, P_i, \dots, P_n)])((g^{a_i})^{\alpha+1}, (g^{a_i})^{\tilde{k}} \cdot g^{k_i}, \\ &\tilde{r}, k_i \tilde{r} \cdot \text{hash}(\tilde{r}, M) \pmod{q}, \tilde{k}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q)), \\ &\text{where } \tilde{r} \in_R \mathbb{Z}_p^*, \text{gcd}(\tilde{r}, q) = 1, k_i \in_R \mathbb{Z}_q^*, M \in_R \{0, 1\}^*, a_{[1,n]} \text{ is appropriate} \\ &\text{for } (P_1, \dots, P_i, \dots, P_n), a_{[1,n,\{i\}]} \in \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*, k_{[1,n,\{i\}]} \in_R \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*, \\ &\alpha \in \mathbb{Z}_q^* \setminus \{q - 1\}, \tilde{k} \in \mathbb{Z}_q^* \setminus \{1\}, \\ &\alpha = (\dots((a_1 + 1)a_2 + 1)\dots)a_{i-1} \pmod{q}, \\ &\tilde{k} = -(\alpha + 1)(\tilde{r} \cdot \text{hash}(\tilde{r}, M))^{-1} \pmod{q}. \end{aligned}$$

Therefore $\mathbf{DLP}_q \equiv_T^p \mathbf{Secret2}[(P_1, \dots, P_i, \dots, P_n)]$.

Security of Signing Order. We study the order of signature creation. Namely we assess the difficulty of forgery when all attackers $P_k \in G_{i,\dots,j}$ excluding P_i try to collude to change the signing order from $G_{i,\dots,j}$ to $G_{j,\dots,i}$, where $1 \leq i < j \leq n$, after P_i , who is given a possibly forged partial signature from the previous signer, has signed M . Attackers can use any information except a_i and k_i .

Let y' be a public key of signature structure $(\dots, P_{i-1}, P_j, \dots, P_i, \dots)$ and (r', s') be a forged signature. \tilde{k} and \tilde{r} are freely selected by attackers.

- $\mathbf{Flip}[G_{i,\dots,j}, G_{j,\dots,i}] \leq_{1-tt}^p \mathbf{DLP}_q$:
- $$\begin{aligned} \mathbf{Flip}[G_{i,\dots,j}, G_{j,\dots,i}] &= (y_{i-1}, y_i, y', \tilde{r}_{i-1}, r_i, \tilde{r}, \tilde{s}_{i-1}, s_i, \tilde{k}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, \\ &M, g, p, q) = (\mathbf{DLP}_q(y', g, p, q) + \alpha' r' \cdot \text{hash}(r', M) \pmod{q}, r'), \\ &\text{where } \alpha' \in_R \mathbb{Z}_q^*, r' = g^{\alpha'}, \text{gcd}(r', q) = 1. \end{aligned}$$
- $\mathbf{DLP}_q \leq_{1-tt}^p \mathbf{Flip}[G_{i,\dots,j}, G_{j,\dots,i}]$:
- $$\begin{aligned} \mathbf{DLP}(g^{a_i}, g, p, q) &= \beta^{-1} \cdot \{-\gamma - \alpha' r' \cdot \text{hash}(r', M) + FC(\mathbf{Flip}[G_{i,\dots,j}, G_{j,\dots,i}] \\ &(g^{\alpha}, (g^{a_i})^{\alpha+1}, (g^{a_i})^\beta \cdot g^\gamma, g^{\tilde{k}}, (g^{s_i} \cdot (g^{a_i})^{-(\alpha+1)})^{(\tilde{r} \cdot \text{hash}(\tilde{r}, M))^{-1}}, \tilde{r}, \alpha + \tilde{k} \tilde{r} \cdot \text{hash}(\tilde{r}, M), \\ &s_i, \tilde{k}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q))\} \pmod{q}, \end{aligned}$$
- where $\tilde{k} \in_R \mathbb{Z}_q^*, s_i \in_R \mathbb{Z}_q, \tilde{r} \in_R \mathbb{Z}_q^*$ with $\text{gcd}(\tilde{r}, q) = 1, a_{[1,n,\{i\}]} \in \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*$ is appropriate for $\{G_{i,\dots,j}, G_{j,\dots,i}\}, k_{[1,n,\{i\}]} \in_R \mathbb{Z}_q^* \times \dots \times \mathbb{Z}_q^*, M \in_R \{0, 1\}^*, \alpha' \in_R \mathbb{Z}_q^*, r' = g^{\alpha'}, \text{gcd}(r', q) = 1,$
- $$\begin{aligned} \alpha &= (\dots((a_1 + 1)a_2 + 1)\dots)a_{i-1} \pmod{q}, \\ \beta &= \alpha a_{i+1} \cdots a_n + a_{i+1} \cdots a_n + a_{i+1} \cdots a_{j-1} a_{j+1} \cdots a_n + \cdots + \\ &\quad + a_{j-1} a_{j+1} \cdots a_n + a_{j+1} \cdots a_n \pmod{q}, \text{gcd}(\beta, q) = 1, \\ \gamma &= a_{j+1} \cdots a_n + \cdots + a_n \pmod{q}. \end{aligned}$$

Therefore $\mathbf{DLP}_q \equiv_{1-tt}^p \mathbf{Flip}[G_{i,\dots,j}, G_{j,\dots,i}]$.

7.2 Appropriate Secret Keys

Given $a_i \in \mathbb{Z}_q^*$ select $a_{[i+1,n]} \in_R \mathbb{Z}_q^* \times \cdots \times \mathbb{Z}_q^*$ after selecting $a_{[1,i-1]}$ appropriately for (P_1, \dots, P_i) . The probability that $a_{[1,n]}$ is appropriate for $(P_1, \dots, P_i, \dots, P_n)$ can be estimated as follows.

Let $\alpha_{i-1} = (\cdots ((a_1 + 1)a_2 + 1) \cdots) a_{i-1} \pmod{q}$. Then a_i should satisfy

$$a_i \neq \alpha_j(\alpha_{i-1} + 1)^{-1} \pmod{q} \quad \text{for } j = 1, \dots, i-1$$

and

$$a_i \neq (a_u - \sum_{l=i+1}^m \prod_{t=l}^m a_t)(\alpha_{i-1} + 1)^{-1} \left(\prod_{k=i+1}^m a_k \right)^{-1} \pmod{q}$$

for $u = 1, \dots, m-1$ and $m = i+1, \dots, n$. Thus the probability is

$$1 - (i-1 + \sum_{v=i}^{n-1} v)/2^{|a_i|} = 1 - (n(n-1) - i^2 + 3i - 2)/2^{|a_i|-1}$$

for $i = 2, \dots, n$. If $n = O(\text{poly}(|a_i|))$ this probability can be estimated to be almost 1.

8 Parallel Multisignature Scheme

The public key of a parallel multisignature scheme is $y = \prod_{i=1}^n y_i \pmod{p}$, where $y_i = g^{a_i}$ are the partial public keys. We must have $y_i \in \mathbb{Z}_p^* \setminus \{1\}$ and the product of any partial keys should not be 1.

Parallel multisignature protocol

1. P_i selects $k_i \in_R \mathbb{Z}_q^*$ and computes $r_i = g^{k_i} \pmod{p}$. P_i repeats this process until $\gcd(r_i, q) = 1$
2. Each P_i broadcasts r_i to all the other signers. Then each P_i checks if there is a combination of r_i whose product is equal to 1 modulo p . If there is such a combination, step 1 is repeated.
3. $r = \prod_{i=1}^n r_i \pmod{p}$.
4. P_i computes $s_i = a_i + k_i r \cdot \text{hash}(r, M) \pmod{q}$, and broadcasts s_i to all the other signers.
5. $s = \sum_{i=1}^n s_i \pmod{q}$.

The multisignature of M by $\wedge(P_1, \dots, P_n)$ is (s, r) . It is verified by checking that

$$g^s \equiv g^{\sum_{i=1}^n s_i} \equiv \prod_{i=1}^n y_i \left(\prod_{i=1}^n r_i \right)^{r \cdot \text{hash}(r, M)} \equiv y r^{r \cdot \text{hash}(r, M)} \pmod{p}.$$

Observe that,

$$\begin{aligned} g^s &\equiv g^{\sum_{i=1}^n s_i} \equiv \prod_{i=1}^n y_i \left(\prod_{i=1}^n r_i \right)^{r \cdot \text{hash}(r, M)} \\ &\equiv yr^{r \cdot \text{hash}(r, M)} \pmod{p}. \end{aligned}$$

As noted in [DOMU94, DOM98] parallel multisignatures schemes are essentially threshold schemes [DF91].

9 Multisignature for a Mixed Structure

The parallel and serial signature structures can be combined in an arbitrary order. The public keys and signatures are generated and verified by using the methods of the serial and parallel cases in a straightforward way.

The partial public keys of the signers in a mixed structure are computed as follows. Let G_{init} be the group of signers with no incoming edges in the graph representation G . The partial public key of signer P_i in G_{init} is simply $y_i = g^{a_i}$. This is given to every signer in $G_{next,init}$, the group of signers whose edges are connected to the edges of G_{init} . Then for all i such that P_i does not belong to G_{init} , P_i 's partial public key is $y_i = (g \prod_{j: P_j \in G_{prev,i}} y_j)^{a_i}$ and is given to every signer in $G_{next,i}$, where $G_{prev,i}$ and $G_{next,i}$ denote a group of signers whose edges are connected to and from P_i in the graph representation, respectively. The public key of the group is $y = \prod_{j: P_j \in G_{last}} y_j$, where G_{last} denotes a group of signers whose edges are combined into the output in the graph representation.

General multisignature protocol

- Computing r :** For all P_i which belong to G_{init} , $r_i = g^{k_i}$, with $k_i \in_R \mathbb{Z}_q^*$. For all P_i that do not belong to G_{init} , $r_i = (\prod_{j: P_j \in G_{prev,i}} r_j)^{a_i} g^{k_i}$ where $k_i \in_R \mathbb{Z}_q^*$. The created r_i is given to every signer in $G_{next,i}$. Then $r = \prod_{j: P_j \in G_{last}} r_j$.
- Computing s :** For all P_i which belong to G_{init} , a partial signature (s_i, r_i, r) of P_i is computed by $s_i = a_i + k_i r \cdot \text{hash}(r, M) \pmod{q}$. Then for all P_i that do not belong to G_{init} , the partial signature (s_i, r_i, r) of P_i is computed by $s_i = ((\sum_{j: P_j \in G_{prev,i}} s_j) + 1)a_i + k_i r \cdot \text{hash}(r, M) \pmod{q}$, and given to every signer in $G_{next,i}$. Finally $s = \sum_{j: P_j \in G_{last}} s_j$.

The multisignature of M by the group G is (s, r) . It is verified by checking that $g^s \stackrel{?}{=} yr^{r \cdot \text{hash}(r, M)} \pmod{p}$.

To show how this is done more explicitly we compute the multisignature of the group $G_{fig1} = [\wedge(P_1, (P_2, P_3, P_4)), \wedge((P_5, P_6), P_7, (P_8, P_9))]$ in Figure 1. The partial public key of P_i in G_{fig1} is computed from the secret key $a_i \in \mathbb{Z}_q^*$ and other partial keys y_j ($j \neq i$) as follows: $y_1 = g^{a_1} \pmod{p}$ for the first

lower branch, $y_2 = g^{a_2} \pmod{p}$, $y_3 = (y_2g)^{a_3} \pmod{p}$, $y_4 = (y_3g)^{a_4} \pmod{p}$ for the first upper branch, $y_5 = ((y_1y_4)g)^{a_5} \pmod{p}$, $y_6 = (y_5g)^{a_6} \pmod{p}$ for the second lower branch, $y_7 = ((y_1y_4)g)^{a_7} \pmod{p}$ for the second middle branch, $y_8 = ((y_1y_4)g)^{a_8} \pmod{p}$, $y_9 = (y_8g)^{a_9} \pmod{p}$ for the second upper branch. Finally the public key y of G_{fig1} is computed by $y = y_6y_7y_9 \pmod{p}$.

- **Computing r :** We have $r_4 = r_3^{a_4}g^{k_4} \pmod{p}$, $r_5 = (r_1r_4)^{a_5}g^{k_5} \pmod{p}$, $r_6 = (r_5)^{a_6}g^{k_6} \pmod{p}$, so $r = r_6r_7r_9 \pmod{p}$.
- **Computing s :** We have $s_4 = (s_3 + 1)a_4 + k_4r \cdot \text{hash}(r, M) \pmod{q}$, $s_5 = ((s_1 + s_4) + 1)a_5 + k_5r \cdot \text{hash}(r, M) \pmod{q}$ and $s_6 = (s_5 + 1)a_6 + k_6r \cdot \text{hash}(r, M) \pmod{q}$, so $s = s_6 + s_7 + s_9 \pmod{q}$.

The multisignature of M by G_{fig1} is (s, r) . It is verified by checking that

$$g^s \stackrel{?}{=} yr^{r \cdot \text{hash}(r, M)} \pmod{p}.$$

9.1 Security Considerations

The partial public key of a signer P_i in a directed series-parallel graph can be expressed as $y = g^{Aa_i+B}$, for some $A, B \in Z_q$. We study the difficulty of signature forgery for a structure G_f , given a partial signature (r_i, s_i) of P_i for the original structure G_o .

- **Forge** $[G_o, G_f] \leq_{1-tt}^p \mathbf{DLP}_q$:
Forge $[G_o, G_f](y_{i-1}, y_i, y', \tilde{r}_{i-1}, r_i, \tilde{r}, \tilde{s}_{i-1}, s_i, \tilde{\beta}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q)$
 $= (\mathbf{DLP}(y', g, p, q) + \beta' r' \text{hash}(r', M) \pmod{q}, r')$, where $\beta' \in_R Z_q^*$, $r' = g^{\beta'}$, $\text{gcd}(g^{\beta'}, q) = 1$.
- **DLP** $_q \leq_{1-tt}^p \mathbf{Forge}[G_o, G_f]$:
 $\mathbf{DLP}_q(g^{a_i}, g, p, q) = \alpha_1'^{-1} \{-\beta' r' \text{hash}(r', M) - \alpha_2' + FC(\mathbf{Forge}[G_o, G_f](g^\alpha, (g^{a_i})^{\alpha+1}, (g^{a_i})^{\alpha'_1} \cdot g^{\alpha'_2}, g^{\beta}, (g^{s_i} \cdot (g^{a_i})^{-\alpha-1})^{(\tilde{r} \cdot \text{hash}(\tilde{r}, M))^{-1}}, \tilde{r}, \alpha + \tilde{\beta} \tilde{r} \cdot \text{hash}(\tilde{r}, M), s_i, \tilde{\beta}, a_{[1,n,\{i\}]}, k_{[1,n,\{i\}]}, M, g, p, q))\} \pmod{q}$,
where $a_{[1,n]} \in Z_q^* \times \dots \times Z_q^*$ is appropriate for $\{G_o, G_f\}$, $G_o, G_f \in \mathcal{G}$, $\tilde{\beta} \in_R Z_q^*$ with $\text{gcd}(g^{\tilde{\beta}}, q) = 1$, $s_i \in_R Z_q$ with $\text{gcd}((g^{s_i} \cdot (g^{a_i})^{-\alpha-1})^{(\tilde{r} \cdot \text{hash}(\tilde{r}, M))^{-1}}, q) = 1$, $\tilde{r} \in_R Z_p^*$ with $\text{gcd}(\tilde{r}, q) = 1$, $M \in_R \{0, 1\}^*$, $k_{[1,n,\{i\}]} \in_R Z_q^* \times \dots \times Z_q^*$, $\alpha = f_\alpha(a_{[1,n,\{i\}]})$, $\alpha'_1 = f_{\alpha'_1}(a_{[1,n,\{i\}]})$, $\alpha'_2 = f_{\alpha'_2}(a_{[1,n,\{i\}]})$, with f_α determined by G_o , $f_{\alpha'_1}$, $f_{\alpha'_2}$ determined by G_f , $P_i \in G_o, G_f$,

Therefore $\mathbf{DLP}_q \equiv_{1-tt}^p \mathbf{Forge}[G_o, G_f]$.

This proof does not hold when: (i) $\alpha'_1 = 0$, (ii) β' satisfies $\text{gcd}(g^{\beta'}, q) = 1$. In both cases we cannot show $\mathbf{DLP}_q \leq_{1-tt}^p \mathbf{Forge}[G_o, G_f]$.

A typical example of the latter case is when the signer structure from the first signer to P_i in G_o is preserved in G_f . For example, it is easy to forge a signature for $G_f = (\dots, P_i, P_{i+2}, P_{i+1}, \dots)$ from an original graph $G_o = (\dots, P_i, P_{i+1}, P_{i+2}, \dots)$.

9.2 Constructible and Available Groups

We give a very rough estimate of the number of signers for which the complete set of signer groups becomes constructible.

Let T_n be the number of all signer structures for n signers whose directed graph G satisfies conditions (G-1) and (G-2). Then,

$$T_n < 2^n(2^n - 1)(2^n - 2) \cdots (2^n - (n - 1)) < 2^{n^2}.$$

A bound for the number K_n of all partial public keys for G is given by,

$$K_n \leq 2(2^n - 1) + 1 = 2^{n+1} - 1.$$

Therefore, the number of all partial public keys for all directed graphs with n signers is bounded by,

$$T_n \cdot K_n < 2^{n^2}(2^{n+1} - 1) = 2^{n^2+n+1} - 2^{n^2}.$$

Using the birthday paradox we can now get an upper bound on the number of signers for which we have constructible groups for any set. We have

$$T_n \cdot K_n < 2^{n^2+n+1} - 2^{n^2} << \sqrt{q}.$$

Thus when $n < \sqrt{\log \sqrt{q}}$ any signer group should have a different public key.

Figure 3 and Figure 4 show directed non-series-parallel graphs whose signer groups are not available. The partial public keys of these structures are the same for any choice of signers' secret keys.

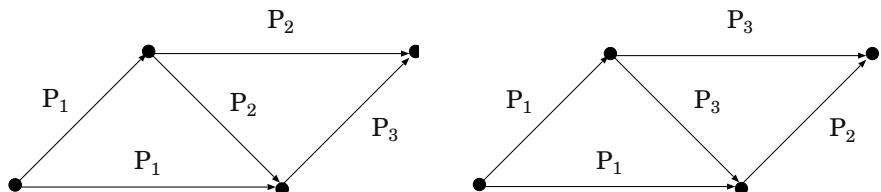


Fig. 3. A non series-parallel graph

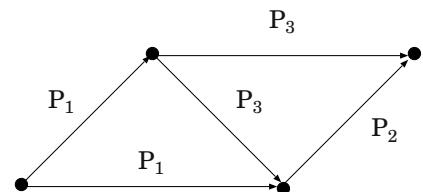


Fig. 4. A non series-parallel graph

Indeed, let $y_{\text{fig.3}}$ and $y_{\text{fig.4}}$ be public keys of these graphs respectively. We can easily check that

$$\begin{aligned} y_{\text{fig.3}} &\equiv g^{((a_1+(a_1+1)a_2)+1)a_3+(a_1+1)a_2} \\ &= g^{((a_1+(a_1+1)a_3)+1)a_2+(a_1+1)a_3} \\ &\equiv y_{\text{fig.4}}. \end{aligned}$$

10 An Efficient Structured Multisignature Scheme

We have not fully exploited the structure of the decomposition tree in our earlier approach. We shall now show how this can be done by considering the group structure $G_{fig1} = [\wedge(P_1, (P_2, P_3, P_4)), \wedge((P_5, P_6), P_7, (P_8, P_9))]$ in Figure 1. From the decomposition tree in Figure 2, using our earlier approach for serial/parallel executions, we see that may take as group secret key, the key

$$a = [[a_1 + ((a_2 + 1)a_3 + 1)a_4] + 1] \cdot [(a_5 + 1)a_6 + a_7 + (a_8 + 1)a_9]$$

with corresponding group public key $y = g^a$. Here a secret key is computed for every sub-tree in Figure 2 precisely once: $[a_1 + ((a_2 + 1)a_3 + 1)a_4]$ is the secret key for the left sub-tree $[\wedge(P_1, (P_2, P_3, P_4))]$. This makes use of $((a_2 + 1)a_3 + 1)a_4$ which is the secret key for (P_2, P_3, P_4) . Similarly $[(a_5 + 1)a_6 + a_7 + (a_8 + 1)a_9]$ is the secret key for the right sub-tree $\wedge((P_5, P_6), P_7, (P_8, P_9))$ which uses $(a_5 + 1)a_6$ as secret key for (P_5, P_6) and $(a_8 + 1)a_9$ as secret key for (P_8, P_9) .

The resulting scheme is far much more efficient than the earlier scheme for which the group key was,

$$\begin{aligned} a' = & (([a_1 + ((a_2 + 1)a_3 + 1)a_4] + 1)a_5 + 1)a_6 + \\ & + ([a_1 + ((a_2 + 1)a_3 + 1)a_4] + 1)a_7 + \\ & + (([a_1 + ((a_2 + 1)a_3 + 1)a_4] + 1)a_8 + 1)a_9. \end{aligned}$$

Since the security for the general case can be reduced to that of the serial and parallel structures, we only need to prove that these are secure. This follows from our earlier reductions.

Let us describe the multisignature protocol for the example of the group structure G_{fig1} with the decomposition tree shown in Figure 2. The partial public key of P_i in G_{fig1} is computed from the secret key $a_i \in \mathbb{Z}_q^*$ and other partial keys y_j ($j \neq i$) as follows: $y_1 = g^{a_1}(\text{mod } p)$ for the first left branch, $y_2 = g^{a_2}(\text{mod } p)$, $y_3 = (y_2g)^{a_3}(\text{mod } p)$, $y_4 = (y_3g)^{a_4}(\text{mod } p)$, for the second left branch, $y_5 = ((y_1y_4)g)^{a_5}(\text{mod } p)$, $y_6 = ((y_1y_4)y_5g)^{a_6}(\text{mod } p)$ for the first right branch, $y_7 = ((y_1y_4)g)^{a_7}(\text{mod } p)$, for the second right branch, $y_8 = ((y_1y_4)g)^{a_8}(\text{mod } p)$, $y_9 = ((y_1y_4)y_8g)^{a_9}(\text{mod } p)$ for the third right branch. Finally the public key y of G_{fig1} for the new protocol is computed by $y = y_6y_7y_9(\text{mod } p)$.

– **Computing r :** We have

$$\begin{aligned} r_4 &= r_3^{a_4}g^{k_4} \pmod{p}, \\ r_5 &= (r_1r_4)^{a_5}g^{k_5} \pmod{p}, \\ r_6 &= ((r_1r_4)r_5)^{a_6}g^{k_6} \pmod{p}. \end{aligned}$$

Similarly,

$$\begin{aligned} r_7 &= (r_1r_4)^{a_7}g^{k_7} \pmod{p}, \\ r_8 &= (r_1r_4)^{a_8}g^{k_8} \pmod{p}, \\ r_9 &= ((r_1r_4)r_8)^{a_9}g^{k_9} \pmod{p}. \end{aligned}$$

Then $r = r_6r_7r_9 \pmod{p}$.

– **Computing s :** We have

$$\begin{aligned}s_4 &= (s_3 + 1)a_4 + k_4r \cdot \text{hash}(r, M) \pmod{q}, \\ s_5 &= ((s_1 + s_4) + 1)a_5 + k_5r \cdot \text{hash}(r, M) \pmod{q}, \\ s_6 &= ((s_1 + s_4) + s_5 + 1)a_6 + k_6r \cdot \text{hash}(r, M) \pmod{q}.\end{aligned}$$

Similarly,

$$\begin{aligned}s_7 &= ((s_1 + s_4) + 1)a_7 + k_7r \cdot \text{hash}(r, M) \pmod{q}, \\ s_8 &= ((s_1 + s_4) + 1)a_8 + k_8r \cdot \text{hash}(r, M) \pmod{q}, \\ s_9 &= ((s_1 + s_4) + s_8 + 1)a_9 + k_9r \cdot \text{hash}(r, M) \pmod{q}.\end{aligned}$$

So $s = s_6 + s_7 + s_9 \pmod{p}$.

The new multisignature of M by G_{fig1} is (s, r) . It is verified by checking that

$$g^s \stackrel{?}{=} yr^{r \cdot \text{hash}(r, M)} \pmod{p}.$$

11 Conclusions

A structured multisignature scheme is a scheme for which the group of signers is structured. We have studied such schemes and proposed structured multisignature schemes based on a modified ElGamal signature scheme. For the security proof we have considered attacks for which all the signers except one honest signer P_i collude. The attackers give a partial signature for a message M to P_i and obtain a valid partial signature for M by P_i . Under this attack model, the security of the proposed scheme is proven by showing reductions of the discrete logarithm problem to the problems of extracting a secret key of a target signer, changing the signing order in a serial multisignature and changing the signing structure in a mixed structure.

Acknowledgements

The authors are grateful to Chandana Gamage and Yuliang Zheng for their kind assistance during the printing trouble of this paper.

References

- [BR93] M. Bellare, and P. Rogaway, Random Oracles are Practical: a paradigm for designing efficient protocols, *Proc. of 1st ACM Conference on Computer and Communication Security*, pp. 62–73, 1993. 468

- [CGH98] R. Canetti, O. Goldreich, and S. Halevi, The Random Oracle Methodology, Revisited, *Proc. of the 30th Annual ACM Symposium on Theory of Computing, STOC*, pp. 209–218, 1998. [468](#)
- [DF91] Y. Desmedt, and Y. Frankel, Shared generation of authenticators and signatures, *Lecture Notes in Computer Science 576, Advances in Cryptology -Crypto '91*, pp. 457-469, 1991. [477](#)
- [DOMU94] H. Doi, E. Okamoto, M. Mambo, and T. Uyematsu, Multisignature Scheme with Specified Order, *Proc. of the 1994 Symposium on Cryptography and Information Security, SCIS94-2A*, January 27-29, 1994. [467](#), [477](#)
- [DOM98] H. Doi, E. Okamoto, and M. Mambo, Multisignature Schemes for Various Group Structures, *The 36-th Annual Allerton Conference on Communication, Control, and Computing*, pp. 713-722, 1999. [467](#), [468](#), [477](#)
- [ElG85] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. on Inform. Theory*, Vol. IT-31, No. 4, pp. 469-472, 1985. [467](#), [472](#)
- [FS86] A. Fiat, and A. Shamir, How to Prove Yourself: Practical Solution to Identification and Signature Problems, *Lecture Notes in Computer Science 263, Advances in Cryptology -Eurocrypt '86*, Springer-Verlag, pp. 186-194 1987. [468](#)
- [HMP94] P. Horster, M. Michels, and H. Petersen, Meta-ElGamal Signature Schemes, *Proc. of the 2nd ACM Conference on Computer and Communications Security*, pp. 96-107, November 1994. [467](#)
- [HMP95] P. Horster, M. Michels, and H. Petersen Meta-multisignature schemes based on the discrete logarithm problem, *Information Security -the Next Decade, Proc. of IFIP/Sec'95*, Chapman & Hall pp. 128-142 1995. [467](#)
- [HZ92] T. Hardjono, and Y. Zheng A practical digital multisignature scheme based on discrete logarithms, *Lecture Notes in Computer Science 718, Proc. of Auscrypt'92*, Springer-Verlag, pp. 122-132, 1993. [467](#)
- [Len90] T. Lengauer, Combinatorial Algorithms for Integrated Circuit Layout, B. G. Teubner Stuttgart, John Wiley & Sons, 1990. [468](#)
- [OO91] K. Ohta, and T. Okamoto, A digital multisignature scheme based on the Fiat-Shamir scheme, *Lecture Notes in Computer Science 739, Advances in Cryptology -Asiacrypt'91*, Springer-Verlag, pp. 139-148, 1993. [467](#), [468](#)
- [OO99] K. Ohta, and T. Okamoto, Multisignature schemes secure against active insider attacks, *IEICE Trans. Fundamentals* Vol. E82-A, No. 1, pp. 21-31, 1999. [467](#)
- [Oka88] T. Okamoto, A Digital Multisignature Scheme Using Bijective Public-Key Cryptosystems, *ACM Trans. on Computer Systems*, Vol. 6, No. 8, pp. 432-441, November 1988. [467](#)
- [PS96] D. Pointcheval, and J. Stern, Security Proofs for Signature Schemes, *Lecture Notes in Computer Science 1070, Advances in Cryptology -Eurocrypt '96*, Springer-Verlag, pp. 387-398, 1996. [468](#)
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communication of the ACM*, Vol. 21, No. 2, pp. 120-126, 1978. [467](#)
- [Schn91] C. P. Schnorr, Efficient Signature Generation by Smart Cards, *Journal of Cryptology*, Vol. 4, No. 3, pp. 161-174 1991. [467](#), [474](#)

- [Shim94] A. Shimbo, Multisignature Schemes Based on the ElGamal Scheme, *Proc. of The 1994 Symposium on Cryptography and Information Security*, SCIS94-2C, January 27-29, 1994. **467**
- [Woll87] H. Woll, Reductions among number theoretic problems, *Information and Computation*, Vol. 72, pp. 167-179, 1987. **470**
- [YL93] S. Yen, and C. Laih, New Digital Signature Scheme Based on Discrete Logarithm, *Electronics Letters*, Vol. 29, No. 12, pp. 1120-1121, 1993. **472**

Author Index

- | | | | |
|----------------|---------------|--------------------|---------------|
| Arita, S. | 58 | Maurer, U. | 93 |
| Banks, W. D. | 68 | Merkle, J. | 14 |
| Bao, F. | 46 | Michałek, D. | 28 |
| Bohannon, P. | 373 | Mizoguchi, F. | 196 |
| Boyd, C. | 223, 433 | Moon, S.-J. | 433 |
| Brickell, E. | 276 | Nieto, J. G. | 223 |
| Buldas, A. | 293 | Okamoto, E. | 466 |
| Burmester, M. | 466 | Okeya, K. | 238 |
| Cant, T. | 75 | Ozols, M. A. | 75 |
| Chan, H. W. | 167 | Park, D. G. | 223, 433 |
| Chong, C. F. | 167 | Pastuszak, J. | 28 |
| Chow, K. P. | 167 | Pieprzyk, J. | 28 |
| Cramer, R. | 354 | Pointcheval, D. | 113, 129, 276 |
| Damgård, I. | 354 | Poupard, G. | 147 |
| Dawson, E. | 223 | Rhee, K. H. | 178 |
| Deng, R. | 46 | Saito, T. | 196 |
| Desmedt, Y. | 466 | Sako, K. | 422 |
| Doi, H. | 466 | Sakurai, K. | 238, 391 |
| Frankel, Y. | 306 | Schoenmakers, B. | 293 |
| Gassko, I. | 342 | Seberry, J. | 28 |
| Gemmell, P. S. | 342 | Shin, S. U. | 178 |
| Henderson, M. | 75 | Shin, W. | 178 |
| Hühnlein, D. | 14 | Shparlinski, I. E. | 68 |
| Hui, L. C. | 167 | Srikwan, S. | 373 |
| Hwang, H. S. | 405 | Stern, J. | 147 |
| Inoue, T. | 391 | Tada, M. | 466 |
| Izu, T. | 210 | Tsang, W. W. | 167 |
| Jakobsson, M. | 373 | Tzeng, W.-G. | 1 |
| Kogure, J. | 210 | Vaudenay, S. | 276 |
| Kohlas, R. | 93 | Verheul, E. R. | 258, 446 |
| Kurumatani, H. | 238 | Wang, X. Y. | 167 |
| Lenstra, A. K. | 446 | Wen, W. | 196 |
| Lieman, D. | 68 | Yokoyama, K. | 210 |
| Lim, C. H. | 405 | Yoshifiji, Y. | 466 |
| Lipmaa, H. | 293 | Young, A. | 326 |
| Liu, C. | 75 | Yung, M. | 276, 306, 326 |
| MacKenzie, P. | 306, 342, 354 | Zhou, J. | 46 |
| Mambo, M. | 466 | | |