

Lecture Notes in Computer Science

1841

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

Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Ed Dawson Andrew Clark
Colin Boyd (Eds.)

Information Security and Privacy

5th Australasian Conference, ACISP 2000
Brisbane, Australia, July 10-12, 2000
Proceedings

Series Editors

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

Volume Editors

Ed Dawson
Andrew Clark
Colin Boyd
Queensland University of Technology
Information Security Research Centre
GPO Box 2434, Brisbane 4000 Queensland, Australia
E-mail: {e.dawson,a.clark,c.boyd}@qut.edu.au

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Information security and privacy : 5th Australasian conference ;
proceedings / ACISP 2000, Brisbane, Australia, July 10 - 12, 2000. Ed.
Dawson ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong
Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1841)
ISBN 3-540-67742-9

CR Subject Classification (1998): E.3, K.6.5, D.4.6, C.2, E.4, F.2.1-2, K.4.1

ISSN 0302-9743

ISBN 3-540-67742-9 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 is a company in the BertelsmannSpringer publishing group.
© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN: 10718964 06/3142 5 4 3 2 1 0

Preface

ACISP 2000, the Fifth Australasian Conference on Information Security and Privacy, was held in Brisbane, Australia, 10–12 July, 2000. The conference was sponsored by the Information Security Research Centre at Queensland University of Technology, the Australian Computer Society, Telstra, Boeing Australia Limited, SecureGate Limited, and RSA Security Pty Ltd. We are grateful to all these organizations for their support of the conference.

The conference brought together researchers, designers, implementors, and users of information security systems. The aim of the conference is to have a series of technical refereed and invited papers to discuss all different aspects of information security. The program committee invited seven distinguished speakers: Mike Burmester, G.R. Blakley, Bob Blakley, Brian Denehy, Roger Lyle, John Snare, and Alan Underwood. Mike Burmester from Royal Holloway College, University of London presented a paper entitled “A Survey of Key Distribution”; G.R. Blakley from Texas A&M University and Bob Blakley from the IBM Tivoli Security Business Unit presented a paper entitled “All Sail, No Anchor, I: Cryptography, Risk, and e-Commerce”; Brian Denehy from SecureGate Limited presented a paper entitled “Secure Networks or Network Security – Approaches to Both”; Roger Lyle from Standards Australia and John Snare from Telstra presented a paper entitled “Perspectives on Australia’s New Information Security Management Standard”; and Alan Underwood from the Australian Computer Society presented a paper entitled “Professional Ethics in a Security and Privacy Context – The Perspective of a National Computing Society”.

There were 81 technical papers submitted to the conference from an international authorship. These papers were refereed by the program committee and 37 papers were accepted for the conference. We would like to thank the authors of all papers which were submitted to the conference, both those whose work is included in these proceedings, and those whose work could not be accommodated.

The papers included in the conference came from a number of countries including 13 from Australia, six from Japan, five from the USA, four from Singapore, three from Korea, two from Greece, and one each from the UK, Germany, Norway, and Yugoslavia. These papers covered topics in network security, public key cryptography, cryptographic implementation issues, electronic commerce, key recovery, public key infrastructure, Boolean functions, intrusion detection, codes, digital signatures, secret sharing, and protocols.

The conference included a panel session entitled “Future Directions in Secure E-Commerce”. This panel was chaired by William Caelli and included leaders in technology, law, and public policy related to the security issues and problems of electronic commerce.

We would like to thank all the people involved in organizing this conference. In particular we would like to thank members of the program committee for their

effort in reviewing papers and designing an excellent program. Special thanks to members of the organizing committee for their time and effort in organizing the conference, especially Ernest Foo, Gary Gaskell, Betty Hansford, Liz Lipowitz, Mark Looi, Lauren May, and Christine Orme. Finally we would like to thank all the participants at ACISP 2000.

May 2000

Ed Dawson
Andrew Clark
Colin Boyd

Australasian Conference on Information Security and Privacy ACISP 2000

Sponsored by
Information Security Research Centre, QUT, Australia
Australian Computer Society
Telstra
Boeing Australia Limited
SecureGate Limited
RSA Security Pty Ltd

General Chair

Ed Dawson *Queensland University of Technology, Australia*

Organizing Chair

Mark Looi *Queensland University of Technology, Australia*

Program Chairs

Colin Boyd *Queensland University of Technology, Australia*
Ed Dawson *Queensland University of Technology, Australia*

Program Committee

Mark Ames	<i>Telstra, Australia</i>
Bob Blakley	<i>Texas A&M University, USA</i>
Mike Burmester	<i>Royal Holloway College, UK</i>
William Caelli	<i>Queensland University of Technology, Australia</i>
Dieter Gollmann	<i>Microsoft Research, UK</i>
Yongfei Han	<i>SecurEworld, Singapore</i>
Wenbo Mao	<i>Hewlett-Packard Laboratories, UK</i>
SangJae Moon	<i>Kyungpook National University, Korea</i>
Winfried Müller	<i>University of Klagenfurt, Austria</i>
Eiji Okamoto	<i>University of Wisconsin, USA</i>
Josef Pieprzyk	<i>University of Wollongong, Australia</i>
Bart Preneel	<i>Catholic University Leuven, Belgium</i>
Steve Roberts	<i>Witham Pty Ltd, Australia</i>
John Rogers	<i>Department of Defence, Australia</i>
Greg Rose	<i>Qualcomm, Australia</i>
Rei Safavi-Naini	<i>University of Wollongong, Australia</i>

VIII Program Committee

Stafford Tavares
Vijay Varadharajan
Henry Wolfe
Yuliang Zheng
Ed Zuk

Queen's University, Canada
University of Western Sydney, Australia
University of Otago, New Zealand
Monash University, Australia
Rotek Consulting, Australia

Referees

Mark Ames
Paul Ashley
G.R. Blakley
Colin Boyd
Mike Burmester
William Caelli
Nathan Carey
Gary Carter
Liqun Chen
Choo Chong Cher
Jong U. Choi
Andrew Clark
Ed Dawson
Ernest Foo
Gary Gaskell
Mickey Gittler
Dieter Gollmann
Juanma Gonzalez Nieto

Yongfei Han
Mark Looi
Greg Maitland
Wenbo Mao
Keith Martin
William Millan
George Mohay
Marco Casassa Mont
SangJae Moon
Yi Mu
Winfried Müller
Eiji Okamoto
Dong-Gook Park
Ji-Hwan Park
Chris Pavlovski
Josef Pieprzyk
Bart Preneel
Jason Reid

Vincent Rijmen
Steve Roberts
John Rogers
Greg Rose
Rei Safavi-Naini
Andrew Salleh
Leonie Simpson
David Soldera
Willy Susilo
Stafford Tavares
Vijay Varadharajan
Kapali Viswanathan
Huaxiong Wang
Henry Wolfe
Chuan Wu
Yuliang Zheng
Ed Zuk

Table of Contents

Network Security I

Protecting Confidentiality against Trojan Horse Programs in Discretionary Access Control System.....	1
<i>Adrian Spalka, Armin B. Cremers, and Hartmut Lehmler</i>	
Towards a New Authorisation Paradigm for Extranets	18
<i>Richard Au, Mark Looi, and Paul Ashley</i>	
Custom Safety Policies in Safe Erlang	30
<i>Lawrie Brown</i>	

Public Key Cryptography

A Proposal of a New Public Key Cryptosystem Using Matrices over a Ring	41
<i>Heajoung Yoo, Seokhie Hong, Sangjin Lee, Jongin Lim, Okyeon Yi, and Maenghee Sung</i>	
Secure Length-Saving ElGamal Encryption under the Computational Diffie-Hellman Assumption	49
<i>Joonsang Baek, Byoungcheon Lee, and Kwangjo Kim</i>	
Efficient Scalar Multiplications on Elliptic Curves without Repeated Doublings and Their Practical Performance	59
<i>Yasuyuki Sakai and Kouichi Sakurai</i>	

Network Security II

High Performance Agile Crypto Modules	74
<i>Chandana G. Gamage, Jussipekka Leiwo, and Yuliang Zheng</i>	
A Three-Party HTTP Proxy to Support Internet Content Regulation	89
<i>Agung Prasetijo, Paul Ashley, Mark Looi, Andrew Clark, and Gary Gaskell</i>	

Cryptographic Implementation Issues

Cryptanalysis of the m -Permutation Protection Schemes	97
<i>Hongjun Wu, Feng Bao, Dingfeng Ye, and Robert H. Deng</i>	
An Implementation of Bitsliced DES on the Pentium MMX TM Processor .	112
<i>Lauren May, Lyta Penna, and Andrew Clark</i>	

Electronic Commerce I

Securing Large E-Commerce Networks	123
<i>Panagiotis Sklavos, Aggelos Varvitsiotis, and Despina Polemi</i>	

Passive Entities: A Strategy for Electronic Payment Design	134
<i>Ernest Foo and Colin Boyd</i>	

Key Recovery

Key Recovery System for the Commercial Environment	149
<i>Juanma González Nieto, Kapali Viswanathan, Colin Boyd, and Ed Dawson</i>	

A Key Escrow Scheme with Time-Limited Monitoring for One-Way Communication	163
<i>Masayuki Abe and Masayuki Kanda</i>	

Public Key Infrastructure

Key Management for Secure Multicast with Dynamic Controller	178
<i>Hartono Kurnio, Rei Safavi-Naini, Willy Susilo, and Huaxiong Wang</i>	

PKI Seeks a Trusting Relationship	191
<i>Audun Jøsang, Ingar Glenn Pedersen, and Dean Povey</i>	

The PKI Specification Dilemma: A Formal Solution	206
<i>Maris A. Ozols, Marie Henderson, Chuchang Liu, and Tony Cant</i>	

Boolean Functions

Iterative Probabilistic Cryptanalysis of RC4 Keystream Generator	220
<i>Jovan Dj. Golić</i>	

Security Weaknesses in a Randomized Stream Cipher	234
<i>Niels Ferguson, Bruce Schneier, and David Wagner</i>	

Two-Stage Optimisation in the Design of Boolean Functions	242
<i>John A. Clark and Jeremy L. Jacob</i>	

Intrusion Detection

A Novel Engine for Various Intrusion Detection Methods	255
<i>Z. Hui and T.H. Daniel Tan</i>	

Codes

Construction and Categories of Codes	266
<i>G.R. Blakley, I. Borosh, and A. Klappenecker</i>	

Digital Signatures I

Cryptanalysis of Polynomial Authentication and Signature Scheme	278
<i>Hongjun Wu, Feng Bao, Dingfeng Ye, and Robert H. Deng</i>	
Secure Transactions with Mobile Agents in Hostile Environments	289
<i>Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos</i>	
A Multisignature Scheme with Message Flexibility, Order Flexibility and Order Verifiability	298
<i>Shirow Mitomi and Atsuko Miyaji</i>	

Secret Sharing I

Light Weight Broadcast Exclusion Using Secret Sharing	313
<i>Natsume Matsuzaki, Jun Anzai, and Tsutomu Matsumoto</i>	
Cheating Prevention in Secret Sharing	328
<i>Hossein Ghodosi and Josef Pieprzyk</i>	
On Multiplicative Secret Sharing Schemes	342
<i>Huaxiong Wang, Kwok Yan Lam, Guo-Zhen Xiao, and Huanhui Zhao</i>	

Digital Signatures II

On the Security of the RSA-Based Multisignature Scheme for Various Group Structures	352
<i>Hiroshi Doi, Masahiro Mambo, and Eiji Okamoto</i>	
Fail-Stop Confirmer Signatures	368
<i>Yi Mu and Vijay Varadharajan</i>	
An Extremely Small and Efficient Identification Scheme	378
<i>William D. Banks, Daniel Lieman, and Igor E. Shparlinski</i>	

Protocols

An Anonymous Electronic Bidding Protocol Based on a New Convertible Group Signature Scheme	385
<i>Kouichi Sakurai and Shingo Miyazaki</i>	
AKA Protocols for Mobile Communications	400
<i>KookHwi Lee and SangJae Moon</i>	

Electronic Commerce II

A Three Phased Schema for Sealed Bid Auction System Design	412
<i>Kapali Viswanathan, Colin Boyd, and Ed Dawson</i>	

An Online Public Auction Protocol Protecting Bidder Privacy 427
Khanh Quoc Nguyen and Jacques Traoré

Secret Sharing II

Algorithms to Speed Up Computations in Threshold RSA 443
Brian King

Sharing Block Ciphers 457
Ernie Brickell, Giovanni Di Crescenzo, and Yair Frankel

Keynote Papers

All Sail, No Anchor, I: Cryptography, Risk, and e-Commerce 471
Bob Blakley and G.R. Blakley

Professional Ethics in a Security and Privacy Context - the Perspective of
a National Computing Society 477
Alan Underwood

Author Index 487

Protecting Confidentiality against Trojan Horse Programs in Discretionary Access Control System

Adrian Spalka, Armin B. Cremers, and Hartmut Lehmler

Department of Computer Science III, University of Bonn
Roemerstrasse 164, D-53117 Bonn, Germany
Fax: +49-228-734 382. adrian@cs.uni-bonn.de

Abstract. Mandatory access control systems (*MAC*) are often criticised for their lack of flexibility, but they protect a system's confidentiality from a wide range of untrustworthy Trojan Horse programs. On the other hand, discretionary access control systems (*DAC*) place no restriction on flexibility. But, at present, they are generally regarded as inherently defenceless against all kinds of untrustworthy programs. We believe that this trade-off is not unavoidable. We show that, for lack of distinction between a user's and a program's trustworthiness, the vulnerability of *DAC* is design-based. On these grounds we present a modified *DAC*. The central idea is the separation of the management of rights from other activities of a user. The resulting system offers the flexibility of *DAC* and the protection of *MAC*.

1 Introduction

The development of multi-user computers created the need for security and protection. Early works on this issue, eg [5] and [6], addressed both its very technical and very general aspects. This, probably, inspired a systematic computer-orientated, or, to be more precise, operating system-orientated dealing with security and protection. In the early seventies, access control models were recognised as an important means of protection.

One of the first works which mentions discretionary access control systems is [3]. This, and other works of this period, illustrate the motivation for the development of *DAC*. It is mainly driven by a practical consideration of the way a computer administers its resources and the way users interact with a computer. The approach is simple. In a first step, determine the resources of a computer that require protection and identify them as protection units. Resources are manipulated and consumed by processes, but each process acts on behalf of a user account. Thus, in the second step, identify the user accounts as the acting or active units. And, lastly, assign to each protection unit a user account, which is said to be its owner. Like the owner of a car can decide where to drive or whom to give a ride, the owner of a resource can decide with whom and in which way he would like to share his resource. Since the way a protection unit will

be shared or not shared cannot be anticipated - it is completely at the owner's discretion -, it is best to provide for all possibilities. The result of this process is the discretionary access control model, a state of which is represented in - and which itself is sometimes colloquially identified with - the well known access control matrix.

At the same time, government and military institutions pursued a different aim. Less interested in the way a computer organises its resources, they desired a protection system that reflects their fixed organisational structure and their way of handling classified documents. The protection objectives were clearly stated by the mandatory security policy, which is aptly explained by [7]:

The dissemination of information of a particular security level (including sensitivity level and any compartments or caveats) to individuals lacking the appropriate clearances for that level is prohibited by law¹.

[1] presented a solution to the problem of enforcing this requirement on computers: the Bell/La Padula mandatory access control model (*MAC*). Like the *DAC*, the *MAC* has protection units, which are denoted as objects, and active units, denoted as subjects. In concordance with the original situation, each object and subject has an additional attribute, its classification. Lastly, the access control rules are described solely by two properties, the Simple-Security-Property and the *-Property, which refer to the classification of objects and subjects.

Today, both models are regarded as fundamentally different, and, to motivate this, the following three reasons are often mentioned².

1. They have little in common with respect to their definition. Both have objects and subjects. But *DAC* uses an access control matrix and *MAC* uses classification based access rules.
2. They have a different degree of flexibility. *DAC* can be adapted to any environment. *MAC* requires the resources and users to be classified or cleared and the classification levels partially ordered³.
3. They have contrasting protection qualities. *DAC* is said to be inherently defenceless against all kinds of untrustworthy programs. Whereas *MAC* provably protects from a wide range of untrustworthy programs.

Thus, *DAC* is said to offer good flexibility but poor protection. In contrast to it, *MAC* offers little flexibility, but good protection. Presented as orthogonal properties, more flexibility is only achievable at the expense of protection, and vice versa. This does not seem to be surprising, and even the historical development can be mentioned in support of it.

Although seemingly accredited, this distinction does not seem to be indisputable. We have grounds to believe that *DAC* can be modified in such a way that *MAC* can be seen as a special-case-secure-DAC.

¹ Landwehr (1981):249.

² There are many works we can cite here. One of the more recent is Castano et al (1994), cf, eg, p 81.

³ Some authors even require the levels to form a lattice.

Such a view is obvious with respect to the security policies. According to a discretionary security policy, each user can determine other users' rights to the objects he owns at his own discretion. In a mandatory security policy we can regard one special user, the trusted labeller, as the owner of all objects, who is allowed - within certain limits - to determine all other users' rights to them at his own discretion. And an ordinary user can only act within the limits set by his security level.

In respect of the above-mentioned first difference in the definitions of the models, it is easy to translate into and represent the classification and the access control rules of *MAC* as an access control matrix. The second difference supports our view. But we can also show [formally, if desired] that each access control matrix can be translated to the *MAC* context, ie be represented as an *MAC*.

The only remaining problem is the third difference. We present a solution to it based on the following observations. Trojan Horse programs, which attempt to violate confidentiality, exploit two possibilities of rights management:

- the free change of rights to existing objects: the malevolent program grants read-access to the object in contravention of the owner's intention
- the free choice of rights to newly created objects: the malevolent program creates a copy of the object or part of it and grants read-access to the copy in contravention of the owner's intention

MAC counters this threat by eliminating both options. An ordinary user (ie not the trusted labeller) cannot change the classification, ie the rights, to an existing object. And, once logged in at a particular security level, the classification of this level, ie the rights corresponding to it, is always assigned to all objects he creates.

To design AS-DAC, the Advanced Security Discretionary Access Control Model, which should protect a system's confidentiality against Trojan Horse programs, we propose to introduce two user-accounts for each user. In the first one, the trusted account, he will act as his own trusted labeller, ie he can change the rights to existing objects owned by him and determine the rights that will apply to objects created in his other account. Since, analogous to *MAC*, all operations the user is permitted to perform in this account must be trustworthy, we assume that these are the only admissible operations in a trusted account. In his other account, the general account, a user can perform any operation except those of the trusted account. The operations admitted here need not be trustworthy with respect to confidentiality, ie they may comprise Trojan Horse programs wanting to compromise the system's confidentiality. The result is an access control model which offers the flexibility of *DAC* and the protection of *MAC*.

Our ideas presented in this paper are already implemented in Linux. We have modified the kernel, the file system and all affected system calls and other parts of the operating system, which a Trojan Horse program can use to compromise confidentiality. The implementation is freely available upon request to the authors.

The subsequent section defines mandatory and discretionary confidentiality policies⁴ and access control models. Section 3 analyses the protection principles of *MAC* from the viewpoint of discretionary controls. On these grounds section 4 introduces AS-DAC. Lastly, a conclusion will summarise the results of this work and present some thoughts on the future development of AS-DAC.

2 MAC and DAC

At the general level, a security policy is an - often informal - assessment of the value of the computer resources and the determination of the resulting protection requirements. A mandatory security policy definitely ascertains both components, a resource's value and the protection requirements. A discretionary security policy only states at whose discretion their determination is left.

The notion of security comprises at least four factors on which it can actually depend: confidentiality, integrity, availability and accountability. However, some prominent security policies care only about confidentiality. We believe that in such cases they should be named confidentiality policies instead.

2.1 Mandatory Confidentiality Policies

Originally, mandatory confidentiality policies were developed to protect paper-based documents of military and government agencies. They rely on a classification scheme that assigns to each document and to each person a security level. The protection requirements are then determined according to the citation of [7] presented on page 2. In addition to it, the following assumption is made about the behaviour of the persons:

When a document is not in a safe, it is in the custody of some individual trusted not to distribute it improperly.⁵

This statement is very important because it clearly shows the grounds on which rights to persons are granted: when A grants to B a right, eg to have read-access to an object, then A assumes that B will not use this right in contravention of A's intentions or interests.

Both statements allow us to view the security level of a document, its sensitivity, as its degree of confidentiality, and the security level of a person, his clearance, as his degree of trustworthiness.

⁴ To be more precise we prefer the word 'confidentiality' instead of 'security'. Confidentiality precisely states its protection objective, whereas security - although often used with the intended meaning of confidentiality - can as well refer to integrity, availability etc.

⁵ Landwehr (1981):250. This statement also demonstrates that the *-Property (introduced on page 6) is not concerned with untrustworthy users but only with untrustworthy programs.

Formally, we therefore define an instance of a mandatory confidentiality policy, *MCP*, as follows:

$$MCP = (S, O, (L \leq), F, PMR)$$

1. *S* is the set of user accounts called subjects (corresponds to persons)
2. *O* is the set of protection units called objects (corresponds to documents)
3. *L* is the set of security levels on which the partial order ‘ $\leq$ ’ is defined
4. $F : S \cup O \rightarrow L$ is the labelling function⁶ which assigns a security level to all subjects and objects
5. *PMR* is the following Primitive Mandatory Requirement: Let $s \in S$ and $o \in O$, then o must be kept secret from s if $F(o) \leq F(s)$ does not hold.

The partial order on *L* can be motivated in two ways. From a practical viewpoint⁷, we can consider an ordered set *K* of generic security degrees and a set *C* of compartments, and define (k, c) , $k \in K$ and $c \subseteq C$, as a security level. Then

$$(k_1, c_1) \leq (k_2, c_2) :\Leftrightarrow k_1 \leq k_2 \wedge c_1 \subseteq c_2$$

is a canonical partial order on the set of all security levels. There is also an axiomatic approach, which we introduce in section 4.

In contrast to an often found one, our definition of an *MCP* comprises an additional component, the Primitive Mandatory Requirement. The reason for its inclusion is the necessity of a clear statement on what the security levels are precisely for. In particular, we would like to emphasise that security levels and access classes are not synonyms. An instance of an *MCP* uses security levels to express only prohibitions of information dissemination, ie confidentiality demands. Of course, the application of an *MCP* to a specific environment, ie, a specific operational translation of it may interpret security levels as access classes or access restrictions - however, not until after it has been explicitly shown that this use is compatible with the intention of the prohibitions.

2.2 Mandatory Access Control

Today’s mandatory access control models mostly rely on the work of [1], which was continued and refined by many authors, eg [4]. The Bell/La Padula mandatory access control model, *BLP*, is an operational translation of an instance of a mandatory confidentiality policy into the context of an operating system abstraction. The enforcement of confidentiality demands is accomplished only with access restrictions. In respect of the preceding discussion of the *PMR*, the assumptions of this model allow the direct identification of security levels with access classes.

The effective protection against many kinds of untrustworthy programs is not a quality of *MCP*. As such, an *MCP* is a declarative statement of demands

⁶ The sets *S* and *O* are obviously disjoint.

⁷ Cf, eg, Landwehr (1981).

which has no notion of operating systems or programs. It is definitely a quality of *BLP*, one of *MCP*'s operational translations.

The following examination of *BLP* is the key to recognising the reasons for the weakness of present discretionary access control models. Starting point of *BLP*:

- An instance $MCP(S, O, (L, \leq), F, PMR)$

Aim of *BLP*:

- Operational interpretation of *MCP* in the context of an operating system abstraction

Assumptions made by *BLP*:

- The general user-accounts constitute the set S
- There is a distinguished trusted user-account t , $t \notin S$, the Trusted Labeller
- The assignment of users to user-accounts is externally (to *BLP*) reliably guaranteed
- The following statements hold for t , the Trusted Labeller:
 - Only t is allowed to change $(L, \leq)$, ie, manipulate the security levels and their order
 - Only t is allowed to change S , ie, add or remove user-accounts
 - Only t is allowed to and obliged to classify the user-accounts, ie, t determines the function $F|_S: S \rightarrow L$
 - Only t is allowed to change the clearance of a subject or the sensitivity of an object, ie, to change the value of $F(s)$ or $F(o)$
- The following statements hold for an $s \in S$, a general user-account:
 - The value of $F(s)$ matches the clearance of the user to whom the user-account s has been assigned
 - $F(s)$ represents the maximum degree of trustworthiness of s , ie

$$Z(s) = \{l \in LI \mid l \leq F(s)\}$$

is the set of his allowed security levels

- The following statements hold for the set O :
 - The objects O are a set of resources of the operating system
 - An object can be created only with the create-function and deleted only with the delete-function
 - An object has a state⁸ which can be viewed only with the read-function and modified only with the write-function
- The following statements hold for processes/programs:
 - All processes/programs t can execute are trustworthy
 - Any process/program an $s \in S$ can execute is possibly untrustworthy
- The following statements hold for sessions:
 - The login-procedure is a trustworthy entry into the operating system

⁸ For example, the state of a file are its name and contents.

- At login-time the user identified with the user-account s chooses at his discretion a value $l \in Z(s)$, which determines the security level of the whole session $A(s, l)$
- The sensitivity l is assigned to every object o created during the session $A(s, l)$, ie, if $\text{create}(o)$ is successfully called during $A(s, l)$, then $F(o) = l$
- Interpretation of the *PMR*: To keep o secret from s , s must not be allowed to view o
- Postulates of *BLP*:
- *Simple-Security-Property*:
During the session $A(s, l)$ the function read is defined only for the set

$$R_{A(s, l)} = \{o \in O \mid F(o) \leq F(l)\} \subseteq O$$

- **-Property*:
During the session $A(s, l)$ the function write is defined only for the set

$$W_{A(s, l)} = \{o \in O \mid F(o) = F(s)\} \subseteq R_{A(s, l)}$$

The satisfaction of the *PMR* with respect to trustworthy operations is guaranteed, almost trivially, by the Simple-Security-Property. The largest set of objects an $s \in S$ can read is

$$R_{A(s, F(s))} = \{o \in O \mid F(o) \leq F(s)\}$$

Since *BLP* treats an object's security level as a part of its name, it is easy to show that a user does not even learn of the existence of objects that should be kept secret from him.

The satisfaction of the *PMR* with respect to untrustworthy operations is guaranteed by several precautions. The **-Property* prevents a malevolent program from directly copying a resource's state, or a part of it, to an object the sensitivity of which contravenes the *PMR*. To block more sophisticated paths, a general user-account possesses no rights management operations. Here, the rights to existing objects cannot be changed at all. And the rights given to a new object are always those fixed to the security level of the session. Thus by selecting the session's security level at login-time the user also chooses the rights he would like to give to other users to objects he newly creates.

The set of rights a user s can choose from is that coupled to the security levels of the set $Z(s)$. Since the user can choose any of these levels as a session's security level at his own discretion, we can say that the set $Z(s)$ represents the discretionary share of *BLP*. There are two mandatory aspects with respect to $Z(s)$. Firstly, the trusted labeller decides (at his discretion) on the clearance of an s , and, thus determines the set $Z(s)$. And, secondly, the decision on the rights coupled to a security level is made in concordance with the partial order of the set of security levels and the Simple-Security-Property.

Before we further exploit these observations, we take a look at today's discretionary security.

2.3 Discretionary Confidentiality Policies

Due to their origin, mandatory confidentiality policies were written down, sometimes with stringent precision, a long time ago. Sadly, the written tracks of discretionary confidentiality policies cannot be found that easily. That notwithstanding, the main two statements of a discretionary confidentiality policy are the following ones:

- Each protection unit is owned by a subject
- The owner of a protection unit decides on its protection requirements at his discretion

Both statements are clear and quite simple. The only question we can still ask, how, when and why does a subject become the owner of an object, has been given very little attention. We have the impression that the answer is often dictated - yet not further contemplated - by the convenience of a particular operational translation. For example, in operating systems one takes it for granted and regards nearly as ‘natural’ that the creator of an object is also its owner.

Given that the issue of who becomes owner of what has somehow been resolved, the second statement provides the information for a formal representation. The idea is simple. Firstly, let the owner of each object state his confidentiality demands. A confidentiality demand is the statement ‘I want to keep o secret from s’. Then collect all these demands. And, lastly, group them according to the subjects mentioned in the demands, viz, list for each subject the objects which should be kept secret from him.

Formally, we define an instance of a discretionary confidentiality policy, *DCP*, as follows:

$$DCP = (S, O, secret)$$

- S is the set of persons
- O is the set of protection units
- $secret : S \rightarrow \mathfrak{P}(O)$ is a function that assigns to each subject $s \in S$ a set $secret(s) \subseteq O$.
- $o \in secret(s)$ if o should be kept secret from s

The function *secret* represents the owners’ discretionary decisions. Since there need not be any functional dependence between their decisions, all we can say about the structure of the result is that it is a set. The last point corresponds to the *PMR* in the mandatory case.

2.4 Discretionary Access Control

We now examine discretionary access controls implemented in many operating systems. From the perspective of a systematic approach one would expect that a discretionary access control model is designed with the aim of being an operational translation of an instance of a discretionary confidentiality policy. This, however, is not so.

Today, an instance of a discretionary access control model is often represented as

$$DAC = (O, S, R, G, Z)$$

such that:

- O is a set of resources
- S is the set of user-accounts
- R is the set of rights
- $G : O \times S \rightarrow \mathfrak{P}(R)$ is a (most often constant) rights-assignment function which determines the rights a user-account has to a resource
- Z comprises the following two access rules:
 - The creator of an object becomes its owner
 - Only the owner of an object is allowed to change the rights-assignment to it

The notion access control matrix stems from the fact that both O and S are finite sets and G can thus be represented in tabular form, which can also be regarded as a matrix.

To reveal the susceptibility of DAC to untrustworthy programs we examine it in a style similar to that of BLP .

Starting point of DAC :

- An instance $DCP = (S, O, \text{secret})$
Aim of DAC :
- Operational interpretation of DCP in the context of an operating system abstraction

Assumptions made by DAC :

- All user-accounts constitute the set S
- There is a distinguished user-account a , $a \in S$, the Administrator
- The assignment of users to user-accounts is externally (to DAC) reliably guaranteed
- Only the Administrator a is allowed to change S , ie, add or remove user-accounts
- Let $s \in S$ be the owner of $o \in O$. Then only s is allowed to change other subjects' rights to o , ie, s determines the function $G|_{\{o\}} : S \rightarrow \mathfrak{P}(R)$
- The following statements hold for the set O :
 - The objects O are a set of resources of the operating system
 - An object can be created only with the create-function and deleted only with the delete-function
 - An object has a state which can be viewed only with the read-function and modified only with the write-function
- The following statement holds for processes/programs:
 - All processes/programs an s E Scan execute are assumed to be trustworthy!!

- The following statements hold for sessions:
 - The login-procedure is a trustworthy entry into the operating system
 - The assignment of rights to an object o created during a session of s is unrestricted, ie, if $\text{create}(o)$ is successfully called during a session of s , then $G \upharpoonright_{\{o\}}: S - \mathfrak{P}(R)$ can be freely chosen!!
 - The rights to existing objects owned by s can be changed during any session of s at his discretion!!
- Interpretation of $o \in \text{secret}(s)$: To keep o secret from s , s must not be allowed to view o

Postulate of *DAC*:

- During any session, s can exercise only the rights stated in G and implied by his ownership

The satisfaction of confidentiality demands stated in *DCP* with respect to trustworthy operations is guaranteed, of course, by this postulate. At the same time, we can clearly see the reason for *DAC*'s problems with untrustworthy programs. *DAC* has no precautions to enforce confidentiality demands in the presence of untrustworthy programs because the possibility of their existence has not been accounted for! We have therefore good grounds to believe that *DAC*'s weakness - at least with respect to those threats *BLP* successfully defends against - is in no way inherent to *DAC*, but simply a matter of *DAC*'s today design.

2.5 Summary

The examination presented above has shown that the strength of *BLP* (and the weakness of *DAC*) relies on the following assumptions:

- There is one account such that all programs/processes that can be executed there are trustworthy
- There are trustworthy rights-management operations
- Untrustworthy programs exploit rights-management operations
- The rights applied to newly created objects are determined at login-time
- Accounts in which possibly untrustworthy programs/processes can be executed have no rights-management operations

Thus, these points must be accommodated in discretionary access controls to make them more resistant to untrustworthy programs.

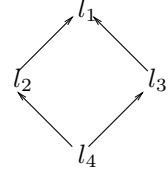
3 A Look at *BLP* from the *DAC*-Perspective

To approach the idea of how we think a stronger discretionary access control model should work, we first present an operational translation of *MCP* in *DAC*-style, however, keeping the above-listed five points in mind.

Let an instance $MCP = (S, O, (L, \leq), F, PMR)$ be given. Throughout this section we illustrate our formal presentation with the following example.

EXAMPLE 1 Let $MCP = (S, O, (L, \leq), F, PMR)$ such that:

- $S = \{s_1, s_2, s_3, s_4\}$
- $L = \{l_1, l_2, l_3, l_4\}, l_4 \leq l_2 \leq l_1$ and $l_4 \leq l_3 \leq l_1$, ie
- $F(s_i) = l_i, i = 1, 2, 3, 4$ $\square$



In the beginning the Trusted Labeller, t , comes into action. Firstly, he assigns to each user his allowed security levels $Z(s) = \{l \in L \mid l \leq F(s)\}$.

EXAMPLE 2 $Z(s_1) = \{l_1, l_2, l_3, l_4\}, Z(s_2) = \{l_2, l_4\}, Z(s_3) = \{l_3, l_4\}, Z(s_4) = \{l_4\}$. $\square$

We now observe that the level at which an object is created is one of its permanent properties, eg a part of its name. For an object $o \in O$ let $\rho(o)$ be the function that returns the security level at which o was created. Then t determines (in discretionary fashion) the session-rights associated with each level in concordance with the PMR . All we need is a rights-mask $C(l)$, for each security level $l \in L$, that determines the access rights associated with this level. If an object is created at level l , then, according to the Simple-Security-Property, only l and greater levels are given the read-right, and, according to the *-Property, only the level itself is given the write-right and the delete-right.

Let $r(l)$ and $w(l)$ be the functions that determine the objects to which a user in a session with the security level 1 has the right to read and to write (we omit delete for it yields the same objects as write), then the general formula for $C(l)$ is:

$$C(l) = \left\{ \begin{array}{l} o \in r(l) \Leftrightarrow l \geq \rho(o) \\ o \in w(l) \Leftrightarrow l = \rho(o) \end{array} \right\} \text{ for all } l \in L$$

From the perspective of discretionary controls, the set of all rights-masks determines the discretionary rights-function G .

EXAMPLE 3 We continue with our example.

- $C(l_1) = \{o \in r(l_1) \Leftrightarrow \rho(o) \in L, o \in w(l_1) \Leftrightarrow l_1 = \rho(o)\}$, ie, It l_1 -sessions have read-access to all objects
- $C(l_2) = \{o \in r(l_2) \Leftrightarrow \rho(o) \in \{l_2, l_4\}, o \in w(l_2) \Leftrightarrow l_2 = \rho(o)\}$, ie, l_2 -sessions have read-access to objects created at the levels l_2 and l_4
- $C(l_3) = \{o \in r(l_3) \Leftrightarrow \rho(o) \in \{l_3, l_4\}, o \in w(l_3) \Leftrightarrow l_3 = \rho(o)\}$ (analogous to the previous point)
- $C(l_4) = \{o \in r(l_4) \Leftrightarrow \rho(o) = l_4, o \in w(l_4) \Leftrightarrow l_4 = \rho(o)\}$, ie, read-access is granted to objects created only at the level l_4 . $\square$

From now on, a user s can login at any allowed level $l \in Z(s)$. The rights to the objects he creates during a session $A(s, l)$ are automatically assigned as specified by the rights mask $C(l)$. Since rights-management operations are unavailable during a session $A(s, l)$, there is no way to manipulate these rights later.

EXAMPLE 4 Suppose that:

- s_1 creates of during the session $A(s_1, l_1)$, ie, at the security level l_1

- s_2 creates o_2 during the session $A(s_2, l_4)$, ie, at the security level l_4

Then the resulting discretionary access control matrix has the following entries:

	l_1	l_2	l_3	l_4
o_1	$\{r, w, d\}$	$\emptyset$	$\emptyset$	$\emptyset$
o_2	$\{r\}$	$\{r\}$	$\{r\}$	$\{r, w, d\}$

We see that the columns of the matrix are labelled with security levels and not with users. Thus a user's rights are not constant but vary according to the security level he chooses for a session.

4 The Advanced Security Discretionary Access Control Model

We are now ready to introduce a discretionary access control model that protects confidentiality against Trojan Horse programs (THP) to the same degree as *BLP* does. The following major modifications and extensions to traditional *DAC* are necessary:

- give to each user two accounts: a restricted one in which he can act only as his own trusted labeller and one for the ordinary work without rights-management operations
- when logging in the ordinary account the user must specify a group of users, the session group, that will have access to the data he is going to create or modify⁹
- the session group is a permanent property of objects created during this session; formally, this feature is an implementation of structured name-spaces in the operating system
- a user's rights depend on the session group specified at login-time

4.1 Basic Notions

We use the following definitions in the remaining part of the paper:

- U is the set of users
- S is the set of work-accounts of U such that for each $u \in U$ there is exactly one $s \in S$
- T is the set of trusted accounts of U such that for each $u \in U$ there is exactly one $t \in T$
- $G = \mathfrak{P}(U)$, the power set of U , is the set of groups of users

⁹ To be precise the data he is going to create or modify must be kept secret from all users not in the session group.

4.2 The Trusted Accounts

In addition to his name and password a user $u \in U$ must specify at login-time either a group $g \in G$, which commences a work-session in his s -account, or the keyword ‘trusted’, which brings him to his trusted account t .

The idea of giving each user an s -account and a t -account is motivated as follows. From the viewpoint of a THP that aims at compromising confidentiality the set of all commands, programs and processes can be partitioned into two subsets: a set, let us say A , that the THP needs to compromise confidentiality and a set, B , that plays no role with respect to it, given that access control rules can be enforced in the s -account.

The t -account comprises only the A -set of commands (and a few helpful but not necessary ones in the strict sense) and it must be given that all commands in this account are trustworthy, ie, there is no way of placing a THP in a t -account. To achieve this in our implementation we have written a special restricted shell in which only the commands listed below can be executed:

- create/modify/delete/list the members of a named group: these commands only enhance the system’s usability; whenever a user needs to specify a group of users he can simply specify the group’s name instead of typing the members
- change the group of a file¹⁰ : this command is critical since only the members of the group have read-access to it; a can change the group of only those files he owns

Remark: our groups are not the Linux-groups; we have a separate administration of groups and the access control system has been modified such that our groups are checked first, before the Linux-groups.

4.3 The Work Accounts

To start a work session a user $u \in U$ must specify his name, password and a group $g \in G$. The choice of g is at the discretion of u . In a declarative manner, a determines with the choice of g that all files he is going to create or modify during this session must be kept secret from all users not in g (thus, $u \in g$ always holds). As we will soon see, the operational translation allows us to regard the choice of g as the granting of read access¹¹ to all its members to all files created or modified during this session.

The three essential components to prevent THPs from doing any damage wrt to confidentiality are: the operation of the create-function, access rules for reading files and those for modifying files.

¹⁰ We use the term file instead of protection object since Linux treats most resources as files.

¹¹ If any other rights are granted as well is irrelevant with respect to confidentiality, but an important question when integrity is at stake. Here we can extend the group with a list of rights that should be granted to the members of g in addition to the read-right.

The adapted create-function has only one essential parameter, the (implicitly full) file name o and no specification of access rights. If successful, it returns the tuple (o, u, g) which states that the file o is owned by u and has been created during a session with the group g . In the s -account there is no way to change u or g ; only u can change g in his t -account. Therefore, it is impossible for a THP to grant any other users access to (o, u, g) than the members of g . We have modified the file system to store this triple in it. It is necessary to ensure that the existence of a confidential file can never be compromised by a naming conflict. Suppose that a user u' has by chance also defined a group with the name g but different members and wants to create a file with the name o . If $u' \in g$ of (o, u, g) , then we can inform him of the file's existence since he has read-access to it anyway. Otherwise we create a new file (o, u', g) . The name-space of files is now structured for, in addition to the file name, it comprises two additional distinguishing components.

We now give the rules for read-access and write-access during a session of u with the session group g , where g denotes the members and not the name of the group, ie, u 's rights vary depending on the choice of g and are limited by the condition that $u \in g$:

1. u has if and only if read-access to (o, u', g') if $g \subseteq g'$, ie u has no read-access if $g' \subseteq g$
2. u has if and only if write-access to (o, u', g') if $g = g'$, ie u has no write-access if $g \neq g'$

Suppose that in u 's g -session there is a THP installed by v , a member of g . The aim of the THP is apparently to get read-access to a file (o, u', g') such that $u \in g'$ but $v \notin g'$. This is precluded by rule (i): read-access is limited to those files which need not be kept secret from any member of the session group. Note: we do not need to consider rule (ii) because u 's decision to select g as the session group reflects his trust in the group's members - to refer to Landwehr - not to distribute files with the g -group improperly.

Now suppose that in u 's g -session there is a THP installed by a user v not in g . The aim of the THP here is to get read-access to a file (o, u', g') such that $u \in g'$ but $v \notin g'$. This is precluded by rule (ii) and our create-function: data read from any accessible file, or any result from it, can only be copied to a file with the g -group, to which v has no read-access (unless the file's owner decides in his t -account to change the file's group).

Until now, these considerations on the resistance of AS-DAC against THPs that try to violate confidentiality are purely informal. We would like to note that we have a full formal logic-based proof of it as well. We gladly include it in this paper should the referees consider it appropriate or necessary.

Lastly, we would like to note that the resistance properties apply only to communications performed with operations intended for this purpose, eg read and write commands. We are very well aware of the fact that other operations or their side-effects can be (mis-)used to establish a communication channel, a so called signalling or cover channel. The answer to the question of how much a handicap this represents is subject to an environment's individual threat assessment.

4.4 Some Remarks

As already mentioned in the introduction, the above-presented concepts have been implemented as a modification of the Linux-kernel. However, there is much more to say to this topic than space would allow for. In particular, we have also examined the following points in detail.

Inter-process communication. THP can try to disclose confidential data by passing them to other processes. Therefore, we have examined the process structure of Linux, the methods to create and destroy processes, and to inherit and to communicate data. Here we have modified and taken control of the process status information, so that THP cannot use it for illegal passing of confidential data.

Remote access and Internet services. The protection against THP is based on groups of known users, ie on closed systems or purely local networks, which de facto are an exception today. Email and the WWW are two ways a THP can use to pass data to any computer in the world. While a user often browses spontaneously in the web, email is most often used with a particular recipient in mind. Thus, different strategies are needed to tame THPs here. Taking THPs at different places into account, we have devised (but not yet implemented) some methods to allow either a risk-free or risk-conscious use of these services.

Role of the administrator. On the way of modifying the system, we have also re-evaluated the administrator's role. In today's systems the administrator has a god-like position, which enables him to do everything at all times - a situation hardly comparable to people with analogous responsibilities in computer-free environments. There they are a kind of porters, caretakers or janitors, ie, people who have no direct power over resources but can only temporarily act on behalf of those who have power. Since we believe that an administrator should play the same role in a computer system we have adapted the procedures (inside and outside the computer) accordingly.

Integrity. With the advent of digital signatures one can easily anticipate that future THPs will more and more attack the data's integrity. Though far from being final, we pursue some promising ideas of how to deal with integrity violations by THPs¹². We follow observations from other fields, which indicate that violation of integrity cannot be prevented but only detected. So we work on detection mechanisms that become effective before the violation is involved in a final act. While this concerns integrity in combination with Internet services, we have implemented provision to ensure integrity locally with the help of rights masks. A user can parametrise a group with a right mask and specify in this way, which rights- in addition to the read-right -the group members should have to files he is going to create.

¹² We concentrate here purely on technical means. At present, similar to issuers of credit cards, issuers of digital signatures try to exclude the users' risk by offering a financial limit on the damage due to unauthorised use. Taken that a digital signature is in fact a digital identity card - and not a digital purse - it is obvious that there is more at stake than just some money.

Shared rights. Soon to be implemented, we have incorporated into our system shared rights. A shared right is given to several users to ensure that important operations cannot be authorised by just a single user, ie a shared right becomes effective only if a certain number of users who have this right decide to exercise it. A simultaneous shared right requires the users participating in the operation to be on-line at the same time, whereas an asynchronous one allows them to authorise the operation at any time.

There are many other details which we have taken care of to make the system work reliably, eg, system calls inspecting and modifying the process, file and directory information, which altogether also contribute to the system's fundamental security properties.

5 Conclusion

A couple of years ago doubts arose if the postulated and seemingly indisputable weakness of *DAC* against THP is inherent to it - a comparison of *MCP* and *DCP* rather hinted that *MCP* should be a special case of *DCP* since *MCP* places a limit on the users' discretionary decisions whereas *DCP* does not. Both *DCP* and *MCP* are declarative definitions, which do not have a notion of a THP. Thus, the suspicion came up that resistance against THP is not a quality of these declarative structures but rather of their operational translations. An evaluation of *BLP* from the perspective of *DAC* revealed that *BLP*'s key to its strength is the distinction made between the trustworthiness of a user and a program started on his behalf - a distinction neglectable in a paper-based world for all human acts are clearly visible. A similar examination of today's *DAC* swiftly revealed that *DAC* does not make this difference, the implicit consequence of which is that all programs are assumed to have the same trustworthiness as the users who started them. This (one can now say, trivial) observation led to AS-DAC.

We have shown (and can provide a formal proof) that AS-DAC is resistant against the same class of THPs and their attacks as is *BLP*. At the same time AS-DAC is a full discretionary control system for it allows the owner of a protection unit to choose at his discretion, from which users this unit should be kept secret and which additional rights should be granted to which users.

The successful completion of the formal modelling motivated us to implement it. Linux, the sources of which are available to the public, is the perfect candidate to test the qualities of AS-DAC. Though the time needed to understand the kernel and the dependencies between the modules and system calls was considerable, we can demonstrate with our implementation of AS-DAC that THPs can very well be tamed in a discretionary access control system.

References

1. Bell, David Elliott, and Leonard J. La Padula.(1975) *Secure computer system: Unified exposition and multics interpretation*. MITRE Technical Report 2997. MITRE Corp, Bedford, MA.

2. Castano, Silvana, Mariagrazia Fugini, Giancarlo Martella and Pierangela Samarati. (1994) *Database Security*. Wokingham, England *et al*: Addison-Wesley.
3. Conway, R. W., W. L. Maxwell and H. L. Morgan. (1972) '*On the Implementation of Security Measures in Information Systems*'. CACM 15.4:211-220.
4. Feiertag, R.J., K.N. Levitt and L. Robinson. (1977) '*Proving multilevel security of a system design*'. *6th ACM Symposium on Operating System Principles. ACM SIGOPS Operating System Review* 11.5:57-65.
5. Graham, Robert M. (1968) '*Protection in an Information Processing Utility*'. CACM 11.5:365-369.
6. Hoffman, Lance J. (1969) '*Computers and Privacy: A Survey*'. ACM CS 1.2:85-103.
7. Landwehr, Carl E. (1981) '*Formal Models for Computer Security*'. ACMCS 13.3:247-278.

Towards a New Authorisation Paradigm for Extranets

Richard Au¹, Mark Looi¹, and Paul Ashley²

¹ Information Security Research Centre, School of Data Communications,
Queensland University of Technology, GPO Box 2434, Brisbane,
Queensland 4001, Australia
{w.au, m.looi}@qut.edu.au

² Tivoli, Security Business Unit, 9020 Capital of Texas Highway,
Building 1, Suite 270, Austin, TX, 78759, USA.
paul.ashley@tivoli.com

Abstract. The development of extranets is transforming enterprise networking. Rather than using proprietary networks to exchange private information, organisations can now set up corporate extranets to exchange data and share applications with strategic partners, suppliers, and customers in a global scale. Because extranets allow third-party users into corporate networks, they need to be extremely secure and external access needs to be highly controllable. Authorisation governs what an entity can do, thus it is a core element in network security. In this paper, we propose a new authorisation framework that can cope with the dynamic and outreaching characteristics of extranets. We apply the technique of one-shot authorisation token in providing extranet users with flexible direct access to applications without authenticating their identities every time. It also solves the problem of revocation and update of user privileges in off-line models. This authorisation scheme has various advantages in terms of higher efficiency and greater adaptability to the diverse application environment of extranets.

1 Introduction

With the increasing use of Internet technology inside the organisation, more computing resources have been connected to networks in the forms of Intranets and extranets. An Intranet is a private, unique network implementation based on the technology, services, protocols and application of the Internet. Its basic purpose is to support an internal information system, and it is usually either physically separate, or protects itself from the outside world through the use of some security system, e.g. firewalls, to restricts the type of traffic or access to a network. Extranets are extended Intranets that connect to outside customers and other more strategic partners. Allowing controlled access by authorised outside parties over the public Internet, extranets enable collaborative business application across multiple organisations. With this potentially global external connectivity, extranets have extra layers of complexity in the terms of functionality and security. Authorisation governs what an entity can do, thus it is a

core element in network security. There are a number of proposed solutions to solve the Intranet authorisation problem. Some leading initiatives are DCE [16] and SESAME [1]. In this paper, we propose an extension to these existing security architectures with a new technique of using privilege credentials in order to tackle the authorisation problem on extranets.

This paper is organised as follows. Some related work about the security issues on extranets and traditional methods of authorisation in distributed systems is discussed. A new framework for providing authorisation services on extranets is proposed. An architectural overview is given and the features of its components are explained. The use of the proposed one-shot authorisation token is described. The paper finishes with a section on future work and conclusions.

2 Related Work

2.1 Security Issues and Challenges with Extranets

Compared with Intranets and proprietary networks used for years, an extranet handles not only the internal world of an organisation but also the cultures of multiple business entities over the public Internet. These give rise to multiple challenges in designing effective access control mechanisms in such an open and dynamic environment. In [6], four security issues unique to extranets are described. These are :

- Shared endpoint security : With an extranet, security becomes the joint responsibility of the organisations at the endpoints of the tunnels that link a group of Intranets or users accessing the network.
- Unmanageable Heterogeneity : An extranet involves a population of local and remote users who may span the globe and multiple organisations. It is virtually impossible to manage or even foresee the types of heterogeneous systems that are used to access the extranet. The role of standards-compliant security becomes important.
- Cross-Pollination : Interaction between multiple systems results in inter flow of data and information. This adds an additional dimension of security to the maintenance of the integrity, accuracy and confidentiality of internal system information.
- Finer Access Granularity : Added levels of access to extranets for multiple business entities based on their partner status means that the mechanisms for controlling extranet access must be constructed with a finer level of granularity.

Since extranet administrators and users must deal with the policies and sensitivity of the organisations being accessed, the following issues are identified:

- Trust Management : Trust establishment becomes a complex task due to the diversity in the extranet environment. Organisational systems must facilitate the ability to access the trustworthiness of entities (users or applications)

they encounter before access rights are granted to them. Thus security tools and protocols for establishing and managing trust effectively across multiple organisations on extranets need to be developed.

- **Security Policy Translation :** Extranets make internal network system available to selected outsiders. It is a fundamental requirement that those external users comply with the security policies in the organisation. There should be some means of translation between different organisations. A corporate extranet establishes linkages between different organisations and allows users to access multiple (both local and external) organisational systems, which may be implementing different sets of security policies. It is a basic requirement that the users have to comply with those policies in their access. Thus new mechanisms for translating and mapping of the security policies from different organisations are required.

Facing all these issues, we require a security solution with access to be highly controllable. And on the other hand, it has to be highly flexible and adaptive to the diverse multi-application environment. In this paper, we propose a new framework for providing authorisation services to the wide spectrum of administrators and users on extranets.

2.2 Traditional Methods of Authorisation in Distributed Systems

User Identity Based Approach. Traditionally, access control adopts the framework of subjects, objects and access rights. While authentication establishes the identities of the subjects (network users), authorisation provides users with certain rights to access objects (services and applications). User authentication provides the mechanism by which access control can be implemented on network data, as well as by which auditing and network monitoring are made easier. In certain environments, establishing a user identity automatically provides the user with a set of privileges. To determine the type of access appropriate for a user, the user's identity is compared to an access control list (ACL). If a user's identity appears on the list, the user is granted the access corresponding to that identity. This identity based authorisation depends on reliable user authentication techniques.

Privilege Certification Approach. An alternative authorisation method is a privilege certification service, the only source of information about a user's privileges that a service provider will trust. The security authority delivers privilege information to the previously authenticated user, encrypted with the provider's key in the form of tickets or certificates. Without the service provider's key, the user cannot read or change the certificate without destroying it. The user forwards the certificate to the service provider, which decrypts and reads it. The key used to encrypt the forwarded message proves to the service provider that the service provided the privilege data and that it has not been modified by the user. The service provider, therefore, trusts the privilege data, conferring only

those permissions that correspond to the certified privileges. Some examples of security architectures using this kind of push technology are

- Kerberos using ticket-granting ticket [12],
- DCE using Privilege Attribute Certificate (PAC) [10] ,
- SESAME using Privilege Attribute Certificate (PAC) [1] and
- Windows 2000 using Security Access Token (SAT) [2].

In these architectures, users have to complete the authentication process each time before privilege certificates are issued to them. We can see that the traditional methods of authorisation directly depend on a reliable user authentication technique.

3 A New Approach for Providing Authorisation Services on Extranets

As a broader definition from ISO [11], authorisation is the act of determining whether an authenticated entity has the right to execute an action. Thus, authorisation can be separated into two stages [2] :

- Granting rights and/or privileges (authorisation credentials) to a particular entity;
- Using these privileges in combination with access decision rules at the resource to determine if access should be granted to the entity.

Hence an authorisation service must include a method for allocating privileges to an entity, and also provide an Access Enforcement Function (AEF) to ensure access is provided only if the privileges satisfy the access decision rules.

We argue that the authentication of a user is essential in the first stage of authorisation but it is no longer needed in the second one. Following this light, we can design schemes that allow users to directly access resources securely with the pre-loaded authorisation credentials and eliminate the repeat of user authentication in every access as in the traditional mechanism. An example is the access control system in hotels. A customer has to complete the registration process before he can get a key to a room for the first time. Afterwards, possession of a key is a sufficient proof to be allowed to enter the room. There is no need for the customer to do registration every time in order to enter the room.

In our scheme, you can prove authorisation without divulging a user's identity in every access. Secure authorisation relies on the security of the authorisation token. The authorisation process is split into two stages :

Stage of User Registration - In the first stage of authorisation, the user has to complete the authentication process before privilege credentials in the form of authorisation tokens are issued. Our approach is similar to DCE and SESAME in the way that a central authorisation server is designed to allocate privileges to authenticated users. It is flexible and adaptive enough to be extended to

use a variety of authentication formats and other distributed information. We propose to place authorisation information such as identity, user groupings and even policy rules into signed credentials that can be distributed at any time with any request by authenticated users. Since they are used for once only, we call them "one-shot" authorisation tokens. In our scheme, we propose the central authorisation server to administer all the applications in its domain (Intranet) as well.

Stage of Access Enforcement - In this second stage of authorisation, every time a user wants to access a service or application, he can submit the authorisation token directly to the appropriate service provider. No user authentication is required. In our authorisation scheme, these authorisation tokens will be handled by the authorisation manager on the application server. The manager evaluates the tokens by an ACL-like mechanism along with whatever access policy information is available, and then grants the appropriate access to the user. The manager is also responsible for the renewal or revocation of the one-shot authorisation tokens, to avoid the repeat of user registration in the next time of access.

4 Architectural Overview

Our proposed framework extends the DCE or SESAME architecture to provide centralised authorisation administration but distributed access enforcement among various service providers in extranet environment. Referring to figure 1, we introduce an Authorisation Server in each Intranet to provide authorisation service to its resources and clients.

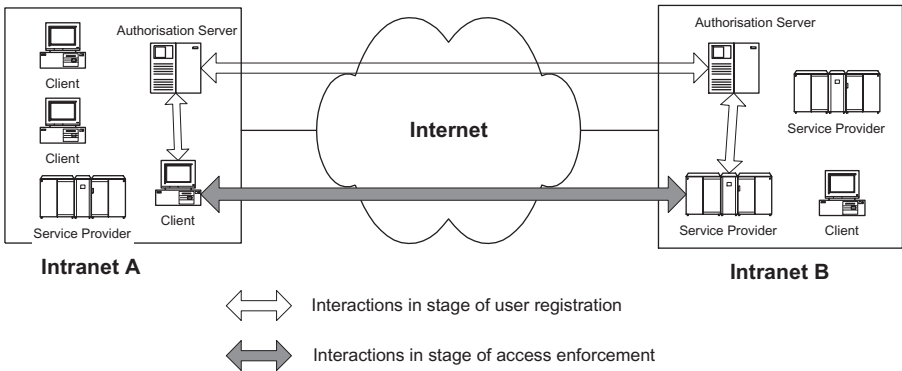


Fig. 1. Authorisation services between two Intranets

4.1 Authorisation Server for Centralised Control

Centralisation offers a cost-effective, easier management structure. It enables the system administrator to manage complex network applications with a high degree of consistency. DCE and SESAME allow central administration of user privileges but the access control rules (normally ACLs) are managed by the administrators of distributed applications. So the authorisation is only partially centrally manageable. In our approach, with the strategy of fully centralised control for an administrator, an authorisation server is designed in each related Intranet to facilitate the following functions :

- To administer the access control rules for all the applications and resources on its local Intranet.
- To create authorisation tokens for accessing local applications or resources and issue them to either local or external authenticated users at the first time of access request (registration).
- To send the updated authorisation information of local users to each application server on the local Intranet or to authorisation servers in other Intranets.
- To establish secure communication channels with authorisation servers on other Intranets.
- To acquire authorisation tokens for local users from authorisation servers on other Intranets.
- To establish mappings and translations for access rules and policies with other Intranets.

4.2 Authorisation Managers for Distributed Applications

While applications and resources are distributed over multiple Intranets, we propose to attach an authorisation manager to each of the application server to facilitate authorisation services. It can be in the form of a program or Application Programming Interface (API). This authorisation manager follows the instructions from the central authorisation server through a secure communication channel. When a user requests access by submitting an authorisation token to the application server, it is the authorisation manager to handle the token and implement the authorisation. The functions of our proposed authorisation manager include :

- Access Enforcement Function (AEF) which makes decisions whether to grant or reject the access request according to the user's privileges submitted and access control information at the resource;
- Token Update Function (TUF) which generates a new authorisation token based on the update information from central authorisation server, if any, and the data on the current token. The update action may be revoking or changing all or part of the current privileges. The renew authorisation token will be sent back to the user later.

- Key management Function (KMF) - It provides session keys to establish a secure channel for the communication between the user and resource or application. These keys may be sent to the user with the renew authorisation token at the same time.

Figure 2 illustrates the system walkthrough for an extranet to be accessed by local and external users. It is highlighted that

- In the stage of user registration, privilege distribution is done only once by the collaboration of the central authorisation servers in related Intranets.
- In the stage of access enforcement, the user make the access request *directly* to the authorisation manager on that related application server. There is no difference in the process of access for local and external users on the extranets.

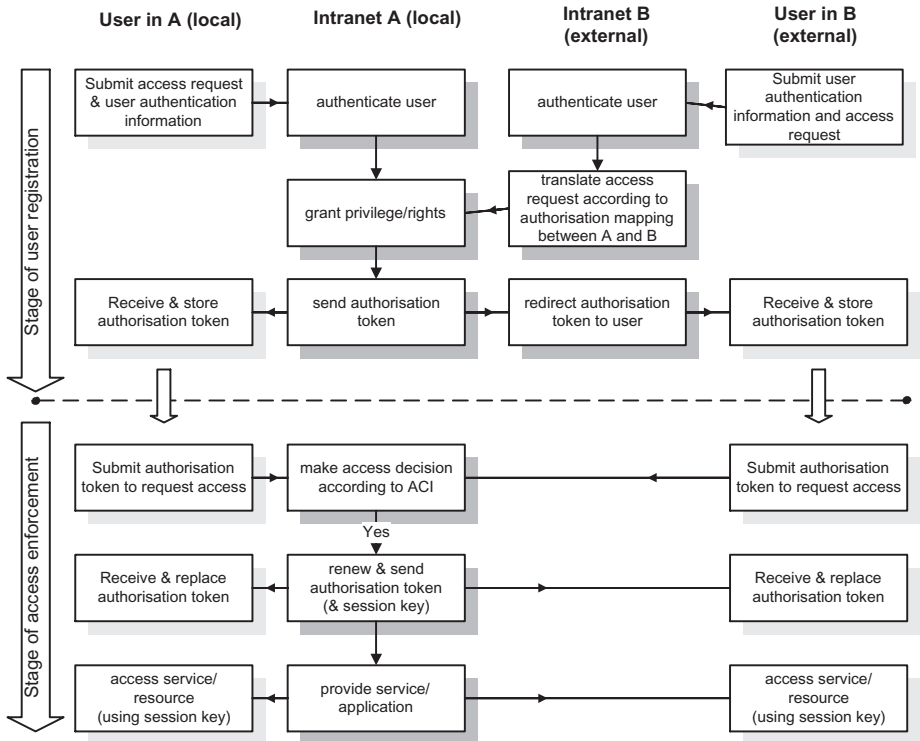


Fig. 2. System Walkthrough for Internal and External Users

5 Use of One-Shot Authorisation Token

An authorisation token is actually a form of digital certificate containing some authorised access control information. Making use of public key and digital signature technology, the authorisation token is issued by the signing authority to delegate some form of authority to the holder of the token. For example, a bank may issue an authorisation token to a customer saying 'the holder is authorised for the withdrawal of \$500 from his account. In general, the owner of any resource that involves electronic access can use authorisation tokens to control the access to it.

In our proposed authorisation framework, we use the one-shot authorisation token to hold the privileges for each individual user on the extranets. As discussed in [5], the token is valid to be used for one time only. It is renewed by the application server and returned to the user in every time of access. This technique provides a mechanism for revocation or updating user's privileges dynamically. Also security can be enhanced as forging and replaying do not work.

5.1 Token Contents

The one-shot authorisation token is a container of access control information for an individual user. In order to suit a wide range of access control policies used on the extranets, the design of the format of the authorisation token has to be modular and flexible. Another important principle is that the authorisation tokens should be designed carefully for specific purposes and the information they contain should be only that required for this and no more. Some common security policies include discretionary access control, mandatory access control, role based access control and task base access control. As an example, an authorisation token may contain access control information (ACI) in one or a combination of the three dimensions, namely identity, task and role :

- **Identity based ACI** : In discretionary access control systems, the access right is based on the identity of user. In these cases, the identity information of the user is included in the authorisation token.
- **Task Based ACI** : An authorisation token may contain identity information that will not generally be necessary and may sometimes be undesirable. For example, 'electronic cash' can be regarded as a specific form of authorisation token in the banking system. Users may want individual transactions to be anonymous. It is possible and necessary to design some authorisation tokens without identity information. To identify the holder of a authorisation token, a bank may typically look up the link between account numbers and owner in its internal databases. Placing such identity information in an authorisation token is actually undesirable since it could expose the holder to additional risks. For example, anyone handling the token may reveal and use the identity information for unintended purposes.

In task based access control, authorisation is done in terms of tasks or activities instead of individual subjects and objects. A task can be defined as a logical unit of work which may involve :

- multiple applications distributed over different Intranets
- multiple subtasks, where a subtask may need to be individually or collectively authorised

Under these policies, the access control information in the authorisation token may contain only the necessary details of the task. It is believed that the authorisation of tasks that span multiple systems over departmental and organisational boundaries can be conceptually simpler and more efficient.

- **Role Based ACI** : Role based access control assign users to roles. The access rights on resources is based on the roles of a user. In this policy, the user are issued with a package of authentication tokens containing the roles information.

In recent research on SDSI, SPKI and PolicyMaker [15] [9] [7], authorisation information such as identity, user groupings and even policy rules is placed into these authorisation tokens that can be distributed at any time with any request. They are evaluated locally by an ACL-like mechanism or evaluation engine, along with whatever local policy information is available.

Attribute certificates as defined in ANSI X9.57 [3] constitute a general-purpose mechanism that bind one or more pieces of attribute information to each certificate's subject. They are further developed in ANSI X9.45 [4] to support the distribution of certified authorisation information.

5.2 Advantageous Features

With the technique of a one-shot authorisation token, the authorisation framework becomes more flexible and adaptive to the needs of extranets. It acquires the following advantages :

- the administration of authorisation is centralised on one security server for each Intranet and this greatly simplifies the management work.
- The access enforcement is done by the authorisation manager which is distributed along with each applications. So the user can use his token to access the application directly. Except for the first time of registration, user authentication to the central security server is eliminated, thus the performance of the system is better as less traffic is possible. Also the problems of bottle neck and single point of failure at the central security server are avoided. This distributed access enforcement system also scales better for large number of users and applications as in the case of extranet.
- The one-shot authorisation token provide a flexible carrier for the mappings of policies and functionality across multiple Intranets in different organisations.
- The mechanism does not interfere with the intra-domain extranet security. Hence it is safer and easier to establish some level of trust among multiple communities.
- While some extranets may use proprietary or exported-restricted cryptographic software or hardware, the format of the token is flexible enough to cater for them individually.

- Protocols using the one-shot authorisation token can be designed to provide mutual authentication between the user and the service provider. It is considered to be an essential assurance of security in the public Internet environment.
- The mechanism can easily be integrated in an existing heterogeneous network because there is no need to modify the internal security architecture of each domain. The one-shot authorisation token can be standardised to become an universal ticket across domains with heterogeneous systems.
- With the use of push technology in our scheme, it is possible for users to access applications anonymously, if the policies allow. Anonymity might be an advantage or necessity in some application system designs, e.g. electronic cash transaction systems.
- Since user log-on procedure is eliminated, access to multiple applications on different extranets becomes easier for individual users. From the view point of the users, it is inefficient and troublesome to replicate the user information on related service providers all over the world. Management of logging on information is not an easy job for individual users if the number of extranets to be accessed is large. Traditional single sign-on for multi-application access on extranet is difficult to be achieved.
- Very often, we need to control access to specific resources from both users and other applications. The use of one-shot authorisation tokens provides a means of authorisation to entities other than users. Also delegation can be achieved by passing these tokens on[5].

6 Future Work

6.1 Use of Smart Card

In our new paradigm, it can be seen that the security relies heavily on the security of the authorisation tokens. Thus a secure personal device, e.g. smart card, is suggested to facilitate as a 'wallet' to hold these tokens safely for individual user. Using the code execution power on smart card, some programs can be developed to work corporately with the authorisation manager on the application server side. This dynamic collaboration can manipulate the authorisation tokens with high security and provide finer granularity of access control [5] in multi-application environment on extranets.

6.2 Trust Management and Security Policy Mappings

On the extranet, multiple business parties across security domains are involved. The creation of the authorisation token requires effective translation of policies and management of trust between those related communities. Some authorisation languages for the development of these mappings of trust, policies and other functionality are required.

6.3 Infrastructure of Authorisation Servers

There would likely be thousands of Intranets with authorisation servers connected together over the Internet. Each authorisation server services a set of local users directly and external users through their authorisation servers. If there are N Intranets joined together to form an enterprise extranet, each authorisation server has to establish some kind of mappings or translations of policies and trust with the other $(N-1)$ authorisation servers. Then there should be $N(N-1)/2$ relationships established over that enterprise extranet. It would be unworkable if N is very large. One solution is to build an infrastructure with tree or hierarchy of the authorisation servers, or perhaps a less structured logical interconnection. Then the initial user registration process and the initial authorisation token delivery can go through a secure chain of authorisation servers. After that, the user can access the application directly with his authorisation token.

6.4 Auditing and Monitoring

Accountability and non-repudiation are two important requirements in the security management. In our authorisation framework, these can only be achieved by the collaboration of the authorisation servers and application servers in the two related domains. Some auditing and monitoring services can be developed based on the trust established and policy mappings among the authorisation servers.

7 Conclusions

We have presented in this paper an authorisation framework for use on extranets. The scheme has the following features :

- Authorisation Service is provided by the collaboration of the central authorisation server and the authorisation managers attached to each distributed application on the extranet. It can enhance the system performance in terms of load balancing and reliability.
- One-shot authorisation tokens provide a user with flexible direct access to applications without revealing the user identity. This technique has various advantages in terms of efficiency, adaptability to the diverse application environment on extranets.

References

1. Ashley P. & Vandenwauver M. (1999). *Practical Intranet Security : An Overview of the State of the Art and Available Technologies*. Kluwer Academic Publishers.
2. Ashley P., Vandenwauver M. & Siebenlist F. (1999). *Applying Authorisation to Intranets : Architectures, Issues and APIs*. Accepted to Computer Communications Journal, Elsevier Science, Vol. 23, 2000.

3. American National Standard X9.57 (1997) *Public Key Cryptography for the Financial Services Industry: Certificate Management*.
4. American National Standard X9.45 (1999) *Enhanced Management Controls Using Digital Signatures and Attribute Certificates*.
5. Au R., Looi M. & Ashley P. (2000) *Cross-Domain One-Shot Authorisation Using Smart Cards*. Unpublished.
6. Bayles D. (1998) *Extranets - Building the Business to Business Web*, Prentice Hall PTR.
7. Blaze M., Feigenbaum J. & Lacy J. (1996) *Decentralised Trust Management*, Proceedings of 1996 IEEE Symposium on Security and Privacy.
8. Butler J.G. (1997). *Securing the Enterprise Network*, Computer Technology Research Corp.
9. Ellison C.M. *SPKI Requirements*, Available at <http://www.ietf.org/internet-drafts/draft-ietf-spki-cert-req-01.txt> on 6-8-99.
10. Harold W. & Lockhart Jr. (1994). *OSF DCE - Guide to Developing Distributed Applications*, McGraw-Hill.
11. ISO 10181-3. (1996). *Information Technology - Open System Interconnect - Security Framework in Open Systems : Access Control Framework*, International Organisation for Standardisation.
12. Kohl J. & Neuman C. (1993) The Kerberos Network Authentication Service V5. RFC1510.
13. Looi M., Ashley P., Seet L., Au R. & Vandenwauver M. (1998) *Enhancing SE-SAMEV₄ with Smart Cards*. Proceedings of conference CARDIS' 98.
14. Nicomette V. & Deswarte Y. (1997). *An Authorisation Scheme For Distributed Object Systems*, Proceedings of Symposium on Security and Privacy IEEE.
15. Rivest R. & Lampson B. *SDSI - A Simple Distributed Security Infrastructure*, Available at <http://theory.lcs.mit.edu/~rivest/sdsi11.html> on 12-9-99.
16. Rosenberry W., Kenney D. & Fisher G. (1992). *Understanding DCE*, O'Reilly & Associates, Inc.

Custom Safety Policies in Safe Erlang

Lawrie Brown

School of Computer Science, Australian Defence Force Academy, Canberra, Australia

`Lawrie.Brown@adfa.edu.au`

Abstract. This paper discusses an approach to specifying a safety policy governing safe execution of mobile code written in the Safe Erlang system. This policy is specified as a check function, which is applied to request messages for access to external resources (using remote procedure calls) from the mobile code executing in a restricted environment. As such, it is similar to the Security Manager used in Java, or the policies defined in SafeTCL. However we believe its application to external resource remote procedure calls by code running in a restricted subnode leads to a simpler and cleaner solution with less overhead; and its specification in a functional language such as Erlang leads to a clearer, more easily verifiable, specification of the policy.

1 Introduction

Mobile code is defined as any code sourced remotely from the system it is executed upon. Because the code is sourced remotely, it is assumed to have a lower level of trust than locally sourced code, and hence needs to be executed within some form of **constrained** or **sandbox** environment to protect the local system from accidental or deliberate inappropriate behaviour.

One of the major issues in supporting mobile code is to provide a suitable level of **System Safety**, which provides appropriate controls on accesses to resources by the executing code so the security of the execution system is not compromised [7,10]. Ideally it would be desirable to be able to formally verify that the imported code will not violate the systems security policy, however in practice this is difficult (cf. for example the work on proof carrying code requiring a hand-crafted proof for each [8]). The other key issue is **Run-Time Safety**, which is concerned with ensuring that the code will behave in accordance with the source specification. This is usually provided by a combination of language features in conjunction with the run-time system.

A number of approaches to providing a safe execution environment have been investigated (see survey by [10]), with much of the focus being on securing procedural languages such as Java [3,9] or Tcl/Tk [4,11]. However here we are interested in how best to extend our existing work on Safe Erlang, which provides the necessary run-time safety, in order to best support a range of **Safety Policies**, and be able to impose differing levels of system safety on different mobile code instances.

Safe Erlang is an extension of the functional language Erlang [6,5]. It includes extensions to Erlang to enhance its ability to support safe execution of remotely sourced code. These extensions are in three areas:

- the provision of a hierarchy of nodes to provide a custom context, restrictions on side-effects (including any external resource access), and resource limits.
- the use of unforgeable references (capabilities) with associated rights of use for modules, nodes, processes, ports, and user defined references.
- support for remote code loading in context.

We have been working with Erlang, rather than the more traditional procedural languages, such as Java or SafeTCL, because we believe that functional languages have a very high degree of intrinsic run-time safety. This means the changes required to provide a safe execution environment are much smaller than those needed to secure a procedural language, as discussed in [6]. Erlang is a functional language, developed by Ericsson, and is currently being used for a number of production telecommunications applications [1,2].

The approach outlined in this paper for specifying a safety policy is based on executing the untrusted mobile code in a constrained environment, which is trusted to prevent direct access to any external resources by the executing untrusted code, and then applying filters to messages between the untrusted code and server processes which mediate the access to resources. Since Erlang generally uses message passing Remote Procedure Calls (RPC's) for servicing requests (including local requests), and since we can enforce this use through our Safe Erlang extensions, this provides a very convenient choke point at which to impose a safety policy. By restricting the problem domain to just verifying that the messages conform to the desired policy, the problem should be considerably more tractable, whilst still being highly extensible. Using Safe Erlang's restricted subnodes, these checks need only be applied to untrusted code running in a restricted subnode. Further, they are done only when the untrusted code is requesting a possibly insecure service provided by a server running in a more trusted subnode. This significantly reduces the overhead of performing such checks. By implementing this approach in a system based on a functional language, it is possible to have the security policy specification written in this language, and then use it directly as the actual check function.

2 Alternate Approaches to Specifying Safety Policies

A range of alternative approaches to imposing safety policies has been tried in a number of languages. In particular the approach used by Java, and that used by Safe/Tcl will be discussed and contrasted with the approach outlined in this paper.

2.1 Java Class Loader, Security Manager, and Access Manager

Probably the best known mobile code system is Java. In Java, system safety is the responsibility of the class loader in conjunction with the security or ac-

cess manager, depending on which version of Java is being used (see detailed discussion of Java security in [9]).

The Java language and run-time system provide a sandbox for code to be executed in, with suitable controls on its operations. The Java Class Loader, Security Manager, and Access Manager interact to determine which particular, potentially unsafe, operations are permitted.

The Java Class Loader is responsible for loading classes as required. It alone knows the origin of the class, whether it has been signed, and hence which security or access manager should be used with it. It is also responsible for managing and enforcing namespace separation rules, which ensures that class/package names sourced from different locations (CODEBASEs) are partitioned into separate namespaces to prevent confusion/subversion due to name collisions. Different Class Loaders are used for code loaded from different sources, and trusted code is permitted to specify the use of a new Class Loader.

The Java Security Manager is called by all Applications Programmer Interface (API) code to see if a requested operation, with the arguments as supplied, is permitted for the requesting class. Only one Security Manager is defined for any instance of a Java run-time system. It is usually selected by trusted code at startup, and cannot subsequently be changed. It includes methods to resolve requests relating to file access, network access, execution of other programs on the local system, and access to system resources. These methods need to be carefully written to handle the varying decisions. They interact with the Class Loader to determine what the requesting class is permitted to do. If different policies need to be applied to different classes, then the Security Manager must have code provided to distinguish between the various cases. In practice, this has proved to be difficult.

Java 1.2 provides a new Access Manager class. It supplements and generalises the features previously provided by the Security Manager (which is maintained to support the current security API). The Access Manager may be consulted by the Security Manager in order to determine whether some action is permitted. In contrast to the Security Manager, customisation of the Access Manager is provided by specifying rules in a policy file, rather than having to modify the actual code. This significantly improves its generality.

The Java approach involves a single execution sandbox. Hence all code running must be vetted for safety - some of the code being permitted to perform "unsafe" operations (because it was sourced locally, or comes from a signed class), whilst other code must be restricted (because it is remotely sourced, untrusted, code). This necessarily complicates the policy specification - in actual code in the case of the Security Manager, or in the policy file for the Access Manager. Also, the realm of what operations are vetted is constrained by which of the API calls include a check with the Security/Access manager as to whether the operation is permitted, although the Access Manager includes an extensible Permissions class, which user applications and classes can choose to use.

2.2 Safe Tcl/Tk Interpreter and Policies

Another well known safe execution environment is that provided by SafeTCL [4, 11]. This involves a master interpreter starting a SafeBase slave interpreter to run any untrusted code. The SafeBase slave interpreter has very restricted functionality (by default, no external access is provided). Potentially "unsafe" operations are supported by extending the slave with additional commands which are actually executed on the master interpreter. These are trusted, and can impose an appropriate safety policy on any requests from the slave. A range of policies can be supported, though on any particular execution, only one is used. This policy specifies when requests for file access, network access, email access, and browser access will be permitted.

Policies are specified in a file, which supplies rules (written as Tcl statements) which govern when each of the categories of potentially "unsafe" operations are permitted, and with what arguments.

The use of multiple interpreters (sandboxes) provides significant flexibility and isolation between the various mobile code modules and the main system. Further the use of a policy specification which contains Tcl commands allows a good degree of flexibility in policy specification, whilst still keeping the policy relatively straightforward. However, Tcl/Tk policies still are bound by the domain of potentially "unsafe" operations which may be vetted.

3 Imposing Policy Using a Message Check Function

The key feature in this proposed approach to providing **System Safety** is the use of a message check function which is applied to request messages for access to resources outside the restricted environment (using remote procedure calls) from the mobile code executing in a restricted subnode. This check function is applied on receipt of the request by the general remote procedure call (RPC) handler for some trusted server executing outside the restricted environment. The message check function acts as both the security policy specification, and the means used to verify that the RPC request messages conform to that policy.

This approach shares a number of similarities with the approach supported by SafeTCL. In both cases multiple execution environments with differing trust levels are supported, along with a flexible policy written in the language used to validate potentially unsafe access requests. Where it differs is in the application of the check to RPC messages, rather than by modifying the code in the API libraries; and in the use of a functional language, which provides a clean specification of the policy.

In Erlang, a general server mechanism is supplied which acts as a wrapper around a number of general services (such as file access). It uses the efficient, lightweight, message passing mechanism supplied by Erlang and used for all inter-process communications. It is quite straightforward to adapt the general server to support the use of a message check function, and then advertising this "protected" service in the restricted environments where unsafe code is

executing. Usually no further changes to the actual server code is required, other than ensuring that the server is started using the modified server mechanism, rather than the default. Thus, unlike other systems, extending the scope of which services are to be protected merely requires that the existing server be started using the new general server mechanism with an appropriate check function, rather than requiring any changes to the actual server code (provided only that the server uses the general server handler).

Writing a suitable check function, in general, defaults to looking at the list of requests supported by the server, and itemising them as always permitted, always denied, or permitted for some parameter values. This then translates fairly directly into the code for the check function. For example, file access may be restricted to files in some specified directory. This requires a policy that includes checking the supplied filename to ensure that it is appropriate.

Whether to impose a policy or not on any particular service is a decision made by the owner of the node. An execution environment can be tailored, comprising one or more restricted subnodes each with a policy imposed by limiting them to using a predefined set of servers, which execute in trusted subnodes, and which have the required check function imposed.

4 Prototype Implementation in Safe Erlang

This approach has been trialed using the SSErl prototype of a safe Erlang execution environment. This system is described in [6]. It is used to create an appropriately tailored execution node for the mobile code, whose only access to resources external to that node is via messages sent to server processes, whose process identifiers are registered by name in the restricted node's names table.

A key feature in implementing this check function is the use of Erlang's pattern matching ability to resolve which of the many possible messages are valid. The check monitor function has the form `check(Mod,Type,Msg)` where `Mod` is the name of the server module the policy is imposed upon, `Type` is the particular RPC message type (call/cast/info) being checked, and `Msg` is the message received by `safe_gen_server` to be relayed to the call-back routines in the actual server module `Mod`. The `Msg` is usually a tuple naming the requested function and providing the necessary parameters.

For example, the monitor function could look as follows:

```
check(file,call,get_cwd) -> ok;
check(file,call,{delete,F}) -> valid_name(F), ok;
check(file,call,{read_file,F}) -> valid_name(F), ok;
check(file,call,{write_file,F,Bin}) -> valid_name(F), ok;
... % other clauses removed
check(file,info,_) -> ok;
check(Mod,Type,Msg) -> exit(policy_violation).
```

This indicates that for the file server, `get_cwd` is ok; as are requests to delete, read or write a file, provided a valid name (say current dir only) is supplied. Info messages (exits etc) are allowed. All other calls will result in a `policy_violation` exception.

4.1 `Safe_gen_server`

Erlang supports a general client-server mechanism to support various services, using the `gen_server` module. With a small change to this module to optionally impose a check function against all messages received for the server, it is possible to run a number of instances of a standard server, each enforcing a different security policy. This results in a clean, and I believe, easier to verify implementation, compared to implementing a suite of custom servers with modified code for each. It also results in the checking overhead only occurring when accessing such servers. In the prototype, this modified general client-server module is called `safe_gen_server`.

`safe_gen_server` is a modified version of the existing `gen_server` module, which implements the security monitor function on messages received for the server it manages. This function is installed by specifying the new `{check, Fun}` option in the call to `safe_gen_server:start(M,A,Opts)` (and related variants). Then, when it receives a message, it will invoke the nominated check function first before invoking the appropriate call-back routine in the server module. If the check function exits, an appropriate error message can be returned to the calling client.

It is left open whether a single check function, covering all modules relevant to a particular safety policy, or whether separate functions for each module, is most appropriate. The current implementation handles both.

To use the check facility, the servers must be started with the check option specified. This would typically require calling `safe_gen_server` explicitly, rather than using the usual start function in the server module. Alternatively, the server module could be modified to provide a variant of `start` for this. The server modules may also need minor changes and recompilation to work in the safe environment anyway (this is certainly true of the prototypes). In either case, the effect is to impose the check function on RPC messages before they are passed to the existing server code.

`safe_gen_server` has also been modified so that an explicit list of node capabilities or global names must be supplied for the `multi.call/cast` interfaces. This is because, with the capability concept, processes must explicitly have capabilities for any resources they wish to use (or know registered names for those capabilities).

4.2 Policy Module

To assist in the creation and usage of safer SSERl nodes, it is desirable that there be a clear and easy mechanism for specifying an appropriate safety policy. This would involve the correct specification of the options in the creation of a new node, but would also require some means of controlling the use of some standard servers, especially those accessing the file system, network and window manager.

For ease of use, policy for an SSErl node may be specified by a policy module in a prescribed form. This module must contain and export a number of functions which are used to create the SSErl node with the desired security policy. These functions are:

- proc_rights()** returns max list of rights for process capabilities, which will be constrained to at most those of the parent node.
- aliases()** returns the list of aliases to be used in the new node. Usually these will map to modified versions of standard library modules.
- init_servers()** calls any server init routines servicing the new node, these servers are run in the current node context, and returns these in the list of registered names to be used when creating the new node.
- check(Mod,Type,Msg)** server message safety check function, which will be called by `safe_gen_server` to check server messages received, when configured in option list (by `init_servers()`).

An example of a policy module is given below. It is intended for use by the SSErl test module as a demonstration. This policy limits access to files to just those in the current directory. More specifically, the functions perform the following:

- proc_rights()** limits the subnode being created to having no special rights, in particular, no right to send external messages, thus limiting message passing to the current node (with its safe servers).
- aliases()** maps the generic module names, especially **file** to the modified, safe, versions (without requiring any change to the code running in the restricted subnode).
- init_servers()** is used to start the file server with the correct policy imposed in the main trusted node, before the restricted subnode is created with this server specified in its namespace.
- check(Mod,Type,Msg)** is the actual check policy imposed on messages received by the file server, specified as a parameter when the server was started.

```
-module(safety_pol).
```

```
% required policy functions we export
```

```
-export([proc_rights/0, aliases/0, init_servers/0, check/3]).
```

```
%% proc_rights is the list of side-effects permitted for process
```

```
%% in node from possible list of: [db,extern,open_port]
```

```
proc_rights() -> [].
```

```
%% aliases is the initial list of module name aliases to use
```

```
%% here alias: the gen_server modules, safe servers, stdlib
```

```
%% modules all of which are compiled for the SSErl env
```

```
aliases() -> [
```

```
    % safe servers
```



```

{file,safe_file},
% extended gen_server modules with support for check function
{gen,safe_gen},{gen_server,safe_gen_server},{proc_lib,
safe_proc_lib},].

%% init_servers calls any server init routines servicing new node,
%%   these servers are run in current node context
%%   returns these in a list of registered names
init_servers() ->
    % start the safe daemons with check function option
    {ok, Filer} = safe_file:start([{check, fun check/3}],
                                  [{safe_file_server,Filer}]).

%% check(Mod,Type,Msg) - server message safety check function
%%   called by safe_gen_server to check server msgs
%%   when configured in option list (by init_servers())
%%   returns ok | {EXIT,{policy_violation,{Mod,Type,Msg}}}
%   handle checks for safe_file server
check(safe_file,call,get_cwd)          -> ok;
check(safe_file,call,stop)             -> ok;
check(safe_file,call,{delete,Nam})     -> valid_name(Nam), ok;
check(safe_file,call,{file_info,Nam})  -> valid_name(Nam), ok;
check(safe_file,call,{read_file,Nam})  -> valid_name(Nam), ok;
check(safe_file,call,{rename,Fr,To})    -> valid_name(Fr),
                                          valid_name(To), ok;
check(safe_file,call,{write_file,Nam,_}) -> valid_name(Nam), ok;
check(safe_file,info,_)                 -> ok;
%   everything else is rejected
check(Mod,Type,Msg) ->
    exit(policy_violation).

... code for valid_name/1 omitted, which checks that
... plain filenames only (no directories) are specified

```

4.3 Creating a Node Implementing a Policy

To create a new node which implements the policy in a policy module, a utility function `safety:polycnode(Parent,Name,Policy)` has been created. Its current implementation is given below.

```

%% polycnode/3 - creates subnode which implements policy module
%% in parent
polycnode(Parent,Name,Policy) ->
    % establish various parameters from policy module
    PRi = apply(Policy,proc_rights,[]), % limit process rights
    Mods = apply(Policy,aliases,[]),   % initial module alias table

```

```
Names = apply(Policy,init_servers,[]),% start servers, get names
% create new node with specified policy
CN = newnode(Parent,Name,[{proc_rights,PRi},{names,Names},
{modules,Mods}])).
```

The `safety:polycynode(Parent,NodeName,PolicyModule)` function may be called as follows to create a custom subnode:

```
PolNode = safety:polycynode(node(),testnode,safety_pol).
```

Once it has been created, processes in it may be spawned as usual, eg. a simple test function in the test module could be run as:

```
spawn(PolNode,test,test,[]).
```

So with this approach, a `SSErl` node with a custom policy may be created by writing an appropriate policy module, and then simply using `polycynode` to instantiate it.

4.4 An Example - `Safe_rpc`

This mechanism has also been used in the `safe_rpc` server. The server executes calls to it in a restricted subnode. The `safe_rpc.policy` module is similar to that shown except that no file server is provided, and the `[extern]` process right is granted, permitting messages to be sent outside the local node. This module is invoked and used by `safe_rpc:start()` as follows:

```
RpcN = safety:polycynode(node(),rpc,safe_rpc_policy),
RpcNode = safety:restrictx(RpcN,[module,newnode]),
Rpc = spawn(RpcNode, safe_rpc, loop, []),
register(safe_rpc, Rpc),
{Rpc,RpcNode};
```

In turn, the `safe_rpc` server itself may be subject to a policy check function, limiting its use.

5 Limitations of the Prototype

There are currently some limitations with the prototyped approach.

The prime limitation is that not all "services" that one might wish to impose a policy upon, use the `gen_server` general server mechanism. Whilst it works for `file` and for `rpc`, other services such as: `pxw`, `sockets`, `unix` do not. Work is progressing on the best approach to handling these.

Also, the `Safe Erlang` system itself is only a prototype, proof of concept. It has shown that the proposed language extension does indeed seem capable of supporting a safe environment, but as it is implemented using a modified compiler as an additional layer above standard Erlang code, it cannot actually provide a completely safe system at present. This will require a new run-time system with internal support for the safety extensions.

The general approach outlined does however, seem to function efficiently.

6 Conclusions

This paper describes an approach to implementing a desired security policy by filtering RPC requests from untrusted (including mobile) code executing in a constrained environment to server processes which provide mediated access to resources outside that environment. This approach provides a simpler and cleaner implementation, compared to other approaches. It differs from the Java approach in that it provides multiple execution environments, each with a potentially distinct policy imposed, rather than having one environment and policy which must handle all of the distinct cases required. It differs from both Java and SafeTCL in that the check is applied against RPC messages sent by code running in an untrusted node and received by trusted servers on which a policy has been imposed, rather than being incorporated in the API and being invoked on every call to the API. Its implementation in a functional language provides a clean and simple specification of the policy, which is then executed as the actual check function. The approach has been trialed using the SSERl safe Erlang prototype, where the code executes in a node whose constraints are specified using a policy module which includes a message check function to check all server messages.

Acknowledgements. The Safe Erlang research was started during my special studies program in 1997, whilst visiting SERC in Melbourne, and NTNU in Trondheim, Norway. I'd like to thank my colleagues at these institutions, and at the Ericsson Computer Science Laboratory in Stockholm for their continuing discussions and support.

References

- [1] J. Armstrong. Erlang - A Survey of the Language and its Industrial Applications. In *INAP'96 - The 9th Exhibitions and Symposium on Industrial Applications of Prolog*, Hino, Tokyo, Japan, Oct 1996. <http://www.ericsson.se/cslab/erlang/publications/inap96.ps>.
- [2] J. Armstrong, R. Virding, C. Wikstrom, and M. Williams. *Concurrent Programming in Erlang*. Prentice Hall, 2nd edition, 1996. http://www.erlang.org/download/erlang_book_toc.html.
- [3] Ken Arnold and James Gosling. *The Java programming Language*. Addison-Wesley, 2nd edition, 1998. ISBN 0201310066.
- [4] Steve Ball. *Web Tcl Complete*. Complete Series. McGraw-Hill, Apr 1999. ISBN 007913713X, <http://www.zveno.com/zm.cgi/in-wtc/>.
- [5] L. Brown. SSERl - Prototype of a Safer Erlang. Technical Report CS04/97, School of Computer Science, Australian Defence Force Academy, Canberra, Australia, Nov 1997. <http://www.adfa.edu.au/~lpb/papers/tr9704.html>.
- [6] L. Brown and D. Sahlin. Extending Erlang for Safe Mobile Code Execution. In *Information and Communication Security*, volume 1726 of *Lecture Notes in Computer Science*, pages 39–53. Springer-Verlag, Nov 1999. <http://www.adfa.edu.au/~lpb/research/sserl/sserl99a.ps>.
- [7] Dan Connolly. Issues in the Development of Distributed Hypermedia Applications, Dec 1996. <http://www.w3.org/OOP/HyperMediaDev>.

- [8] G.C. Necula and P. Lee. Research on Proof-Carrying Code on Mobile-Code Security. In *Proceedings of the Workshop on Foundations of Mobile Code Security*, Monterey, 1997. <http://www.cs.cmu.edu/~necula/fmcs97.ps.gz>.
- [9] Scott Oaks. *Java Security*. O'Reilly, 1998. ISBN 1565924037.
- [10] Tommy Thorn. Programming Languages for Mobile Code. *ACM Computing Surveys*, 29(3):213–239, Sept 1997.
- [11] J. A. Zimmer. *Tcl/Tk for Programmers*. IEEE Computer Society Press, Sept 1998. ISBN 0818685158, <http://www.mapfree.com/sbf/tcl/book/home.html>.

A Proposal of a New Public Key Cryptosystem Using Matrices over a Ring

Heajoung Yoo¹, Seokhie Hong, Sangjin Lee, Jongin Lim, Okyeon Yi², and
Maenghee Sung^{3*}

¹ Center for Information and Security Technologies(CIST),
Korea University, Seoul, Korea,
hjyoo@gauss.korea.ac.kr

² Electronics and Telecommunications Research Institute(ETRI), Korea

³ Korea Information Security Agency(KISA), Korea

Abstract. In this paper, we propose a new public key cryptosystem based on a probably subexponential time problem over a modular matrix ring. Our new cryptosystem is similar to GGH(O. Goldreich, S. Goldwasser, and S. Halevi) cryptosystem but the attack which breaks the GGH system is not applicable to ours. The security of the new construction is based on the computational difficulty of integer factorizations.

1 Introduction

The need for public key cryptosystem is spreading rapidly today as more people use computer networks. In the light of the importance of public key cryptosystem, there are relatively few proposals of public key cryptosystems which have received any attention. Among them, there were two lattice-based public key cryptosystems which were given wide attention: the Ajtai-Dwork cryptosystem[1] and the GGH cryptosystem[2]. Though the security base(the closest vector problem and the shortest vector problem) were NP-hard, they were completely broken since they had major flaws in the design of schemes.

This paper proposes a public key cryptosystem which runs fast and has high level of security. Our encryption scheme is very similar to GGH cryptosystem but ours is not broken by the methods which break GGH.

In 1997, O. Goldreich, S. Goldwasser and S. Halevi proposed a lattice-based public key cryptosystem based on the NP-hardness of CVP [2]. But in 1999, P. Nguyen broke it by observing that ciphertext leaked information on the plaintext and the decryption could be reduced to a special closest vector problem, which was much easier than the general problem[3].

The basic idea of our new cryptosystem starts from fixing the weakness of GGH, so appearance of our system is similar to GGH. But contrary to GGH, the security of our system is based on the RSA problems not on the lattice problems. Computational complexities for encryption and decryption are given as $O(n^2)$, where n is the dimension of the lattice.

* This paper supported by Korea Information Security Agency(KISA) grant 99-S-087.

The rest of the paper is organized as follows: we first review necessary material about the lattice in section 2 and briefly describe the GGH[2] and P. Nguyen's attack in order to compare with GGH[3]. In section 3, we introduce the new public key cryptosystem and in section 4, we suppose various analyses for the proposed scheme and describe conditions of parameters for the security of the proposed scheme. Section 5 presents the results of the computer simulation and the comparison with RSA. Lastly, we discuss how to improve the security of our system in section 6.

2 Background on the New Scheme

In the sequel, we denote vectors by bold-face lowercase letters(e.g. $\mathbf{b}$, $\mathbf{c}$, $\mathbf{r}$) and use capital letters(e.g. B , C , R) to denote matrices or sets of vectors.

2.1 Definitions

Let B be a non-singular $n \times n$ integral matrix. Denote by $\mathbf{b}_1, \dots, \mathbf{b}_n$ the column vectors of B . The lattice L spanned by $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ is the set $L(B)$ of all integral linear combinations of the $\mathbf{b}_i$'s. The set of $\mathbf{b}_i$'s is called a basis of L .

There are two NP-hard problems in the lattice(The shortest vector problem (SVP)[4] and the closest vector problem(CVP)[5]). No polynomial-time algorithm is known for approximating these problems[6].

2.2 The GGH Cryptosystem

We make a brief description of the GGH. The public key and the private key are two representations of the same lattice. A lattice L is generated by a reduced basis R , and then R , which is kept private, is transformed to a non-reduced basis B , which will be public. In GGH, the security parameter is $(n, \sigma) \in \mathbb{N}^2$. A typical value is $(n, \sigma) = (300, 3)$. A message $\mathbf{m} \in \mathbb{Z}^n$ is encrypted into $\mathbf{c} = B\mathbf{m} + \mathbf{e}$, where $\mathbf{e}$ is an error vector uniformly chosen from $\{-\sigma, \sigma\}^n$. A ciphertext $\mathbf{c}$ is decrypted as $B^{-1}R\lfloor R^{-1}\mathbf{c} \rfloor$, where $\lfloor R^{-1}\mathbf{c} \rfloor$ denotes the vector in $\mathbb{Z}^n$ which is obtained by rounding each entry in $R^{-1}\mathbf{c}$ to the nearest integer.

Theorem 1 *Let R be a private basis and denote the maximum l_∞ norm of the columns in R^{-1} by $\gamma/\sqrt{n}$. Then the probability of decryption errors is bounded by*

$$Pr[\text{decryption error using } R] \leq 2n \cdot \exp\left(-\frac{1}{8\sigma^2\gamma^2}\right)$$

Proof See [2].

This theorem implies that the GGH is decrypted without errors with high probability.

2.3 Phong Nguyen's Attack

As seen above subsection, GGH has a typical type of error vector which is much smaller than lattice vector. Let $\mathbf{s} = (\sigma, \dots, \sigma)^T \in \mathbb{Z}^n$. Then $\mathbf{e} + \mathbf{s} \equiv 0 \pmod{2\sigma}$, so that:

$$\mathbf{e} + \mathbf{s} \equiv B\mathbf{m} \pmod{2\sigma}$$

Therefore the attack on GGH scheme is reduced to the general problem of solving a modular linear system. For any ciphertext $\mathbf{c}$, the underlying linear system has very few solutions. Thus, the GGH is far from being semantically secure. If one gets the plaintext $\mathbf{m} \bmod{2\sigma}$ i.e., $\mathbf{m}_{2\sigma}$ (it's possible with high probability), this partial information simplifies the decryption problem as follows;

$$\begin{aligned} \mathbf{c} &= B\mathbf{m} + \mathbf{e} \\ \mathbf{c} - B\mathbf{m}_{2\sigma} &= B(\mathbf{m} - \mathbf{m}_{2\sigma}) + \mathbf{e} \\ \frac{\mathbf{c} - B\mathbf{m}_{2\sigma}}{2\sigma} &= B\mathbf{m}' + \frac{\mathbf{e}}{2\sigma}, \end{aligned}$$

where $\mathbf{m}' \in \mathbb{Z}^n$ such that $\mathbf{m} - \mathbf{m}_{2\sigma} = 2\sigma\mathbf{m}'$. The error vector $\frac{\mathbf{e}}{2\sigma} \in \{\pm\frac{1}{2}\}^n$ is much smaller than previous one. This means that the problem of decrypting ciphertexts could be reduced to the simpler CVP. Applying BKZ reduction algorithm[7], an attacker can find the error vector used in the encryption process.

3 The Proposed Public Key Cryptosystem

1. proposed scheme

Let n be the dimension of a lattice. The basic steps to choose parameters run as follows:

C1 choose positive integers $m, e, d_{ii}, 1 \leq i \leq n$, primes p, q with size 512 bits and matrix $D \in \text{Mat}_n(\mathbb{Z})$ with following conditions;

C1.1 $N = p \cdot q$.

C1.2 m, e : random integers such that $m \approx q^{0.4}, e \approx q^{0.3}$, where m and e are upper bounds of message and error vectors, respectively.

C1.3 D : diagonal matrix such that $m < |d_{ii}| < q^{0.5}$, where $d_{ii}, 1 \leq i \leq n$ are diagonal entries of D .

C2 choose an invertible matrix $T = (t_{ij})_{1 \leq i, j \leq n} \in \text{Mat}_n(\mathbb{Z})$ such that $\sum_{j=1}^n t_{ij} < q^{0.2}$.

C3 $R = D \cdot T$. We can sure that $m + n \cdot l_\infty(R) \cdot e < q$, where $l_\infty(R) = \max_{1 \leq i, j \leq n} \{|r_{ij}|\}$ if $R = (r_{ij})$.

C4 $B = B_q \cdot U \cdot L \pmod{N}$ where $B_q \equiv R^{-1} \pmod{q}$, L (resp. U) $\in \text{Mat}_n(\mathbb{Z})$ is a unimodular lower(resp. upper) triangular matrix whose all entries except the diagonal entries are multiples of q .

public information: B, e, m, N

secret information: R, q, T

2. encryption

Let $M = (m_1, \dots, m_n)^t$, $0 \leq m_i < m$, be a message vector and $E = (e_1, \dots, e_n)^t$, $0 \leq e_i < e$, be an arbitrary error vector. Then the ciphertext is

$$C = B \cdot M + E \pmod{N}.$$

3. decryption

At first, compute $X = (x_1, \dots, x_n)^t$:

$$C_q \equiv C \pmod{q}$$

$$X \equiv R \cdot C_q \pmod{q}.$$

Then $m_i \equiv x_i \pmod{d_{ii}}_{1 \leq i \leq n}$ by the above conditions. The mathematical proof of the decryption procedure is as follows:

$$\begin{aligned} C &= B \cdot M + E \\ &\equiv (B_q \cdot U \cdot L) \cdot M + E \pmod{q} \\ &\equiv B_q \cdot M + E \pmod{q} \end{aligned}$$

$$\begin{aligned} R \cdot C &= R \cdot (B \cdot M + E) \\ &\equiv R \cdot (B_q \cdot M + E) \pmod{q} \\ &= R \cdot (B_q \cdot M) + R \cdot E \\ &= (R \cdot B_q) \cdot M + R \cdot E \\ &\equiv M + R \cdot E \pmod{q} \\ &= M + (D \cdot T) \cdot E \quad \text{by } \mathbf{C3} \\ &\equiv (m_i \pmod{d_{ii}})_{1 \leq i \leq n} \quad \text{by } \mathbf{C3} \\ &= M \quad \text{by } \mathbf{C1}. \end{aligned}$$

4. example of encryption and decryption

Choose $q = 10570841$ and $p = 10570837$. Then $N = 111742637163917$.

And let $m = 500$, $e = 128$, $D = \begin{pmatrix} 612 & 0 & 0 & 0 \\ 0 & 681 & 0 & 0 \\ 0 & 0 & 697 & 0 \\ 0 & 0 & 0 & 601 \end{pmatrix}$, and

$$T = \begin{pmatrix} 5 & 2 & 3 & 7 \\ 4 & 3 & 1 & 2 \\ 4 & 7 & 1 & 3 \\ 2 & 3 & 4 & 9 \end{pmatrix}.$$

Then $R = D \cdot T = \begin{pmatrix} 3060 & 1224 & 1836 & 4284 \\ 2724 & 2043 & 681 & 1362 \\ 2788 & 4879 & 697 & 2091 \\ 1202 & 1803 & 2404 & 5409 \end{pmatrix}$, where q , T , and R are private

keys.

Next, choose

$$U = \begin{pmatrix} 1 & -10570841 & 10570841 & -10570841 \\ 0 & 1 & 10570841 & -10570841 \\ 0 & 0 & 1 & 10570841 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and}$$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 10570841 & 1 & 0 & 0 \\ 10570841 & -10570841 & 1 & 0 \\ -10570841 & 10570841 & 10570841 & 1 \end{pmatrix}.$$

Since $B_q \equiv R^{-1} \pmod{q}$ and $B = B_q \cdot L \cdot U$, then

$$B_q = \begin{pmatrix} 5930539 & -5555340 & -7048431 & 411521 \\ -694743 & 5015501 & -1757594 & 10308964 \\ 4516652 & 3013095 & -1811759 & -2955469 \\ -7791869 & 8794374 & 1782871 & -9523333 \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 53157646921444 & 21634362201988 & 29648146983520 & 33268625202608 \\ 59321909946061 & 30416246925353 & 9377930406397 & 47489354938849 \\ 7621834577836 & 8990344720980 & 100334213765523 & 12995059581902 \\ 104368674961473 & 65200385256960 & 54666820601664 & 8249115384473 \end{pmatrix},$$

where m , e , $n = 4$, and B are public keys.

Let $M = \begin{pmatrix} 151 \\ 81 \\ 259 \\ 462 \end{pmatrix}$ and $E = \begin{pmatrix} 111 \\ 82 \\ 120 \\ 109 \end{pmatrix}$. In this case, ciphertexts are

$$C = \begin{pmatrix} 87570246808078 \\ 32646467360507 \\ 11374775095066 \\ 12518598246761 \end{pmatrix}.$$

Decryption of ciphertexts is as follows:

$$\begin{aligned} R \cdot C &\equiv \begin{pmatrix} 3060 & 1224 & 1836 & 4284 \\ 2724 & 2043 & 681 & 1362 \\ 2788 & 4879 & 697 & 2091 \\ 1202 & 1803 & 2404 & 5409 \end{pmatrix} \cdot \begin{pmatrix} 4613066 \\ 10558157 \\ 495334 \\ 5796624 \end{pmatrix} \pmod{10570841} \\ &= \begin{pmatrix} 1127455 \\ 700149 \\ 1021364 \\ 1159791 \end{pmatrix} \pmod{10570841} \end{aligned}$$

$$\equiv \begin{pmatrix} 151 \pmod{612} \\ 81 \pmod{681} \\ 259 \pmod{697} \\ 462 \pmod{601} \end{pmatrix}.$$

With these procedures, we get a plaintext M .

4 Security Analysis

In this section we provide some analyses for the security of the proposed scheme by considering several possible attacks and describe conditions for security parameters of the proposed scheme.

1. brute-force attack

We do an exhaustive search for the message vector M and the error vector E . Choose an arbitrary vector M' and then let $E' = (e'_1, \dots, e'_n)^t = C - B \cdot M' \pmod{N}$. Recall that $e \approx q^{0.3} \approx 2^{153}$. If $e'_i \leq 2^{153}$ for all $1 \leq i \leq n$, we can accept M' as a message vector with high probability. But the number of possible message vectors is $m^n \approx (q^{0.4})^n \approx 2^{204n}$, so it's impossible to find a message vector by brute-force attack. Alternatively choose a random error vector E' . Since the error bound e is larger than the message bound, it's also impossible to find the error vector by exhaustive search.

2. application of the BKZ-reduction algorithm

The BKZ-reduction algorithm is the most efficient algorithm to solve SVP [7]. As we know, this algorithm can be applicable to the normed spaces. Since our scheme is encrypted by modular operations, this algorithm is not applicable to find error vectors. But even if our space is a normed space, the BKZ-reduction can not break our system as our error vector is much larger than that of GGH.

3. factorization of N

If q is revealed, one can find B_q and D , so our system can be totally broken. But integer factorization is computationally difficult. Hence there is no way to find q .

4. randomness of an inverse matrix modulo q

If n random elements $a_1, \dots, a_n$ are given, the probability of $\text{GCD}(a_1, \dots, a_n) = p$ is $(1/p)^n$. Since i^{th} row of R is a multiple of d_{ii} and B_q is the inverse of $R \pmod{q}$, B_q is possible to leak any information about d_{ii} for some i . We tested whether GCDs of all components of each row in B_q might reveal the information of d_{ii} for some i or not. Our simulation shows that GCDs of all components of in B_q are almost 1 with high probability if n is greater than 12. Hence there is no way to get an information of d_{ii} or t_{ij} from B .

5 Results of a Computer Simulation

In this section, we show results of our computer simulation. Until now, the security of our system based on integer factorization problems, so we compare the efficiency with RSA. The parameters of the proposed scheme is same as those of section 3. One matrix-vector multiplication and one vector addition are needed to encrypt and one matrix multiplication and n modular reductions are needed to decrypt, so the complexities of encryption and decryption are $O(n^2)$. In RSA scheme, we set sizes of two large primes p , q to 512 bits.

	Encryption	Decryption	Platform	Data Size(bits)
RSA	0.13sec	0.19sec	PenII, 333MHz	1,024
Proposed System($n=10$)	0.01sec	0.01sec	PenII, 350MHz	2,048
Proposed System($n=30$)	0.09sec	0.15sec	PenII, 350MHz	6,144
Proposed System($n=50$)	0.27sec	0.43sec	PenII, 350MHz	10,240

Average running time of encryption and decryption

Simulation under above settings shows that running speed of the proposed scheme is faster than RSA in encryption and decryption.

Remark *We implemented the simulation by C-language and this is not optimal.*

6 Conclusion and Further Research

We proposed a new type of public key cryptosystem. For security parameters n , l where n is the dimension of a public matrix and l is the bit length of $N = p \cdot q$, the public key size of the new scheme is $O(n^2 \cdot l)$ and the computation time is $O(n^2 \cdot l^2)$, where the public key size of the l -bit RSA is l and computation time is $O(l^3)$. The further research is; as other systems using a matrix as a public key, the public key size of our system is large, so a major improvement to reduce the public key size is needed to be practical system. And we are studying the difficulty of solving the modular equations since it makes the security basis of our system to be substantial.

References

1. M. Ajtai and C. Dwork, *A Public-key Cryptosystem with Worst-case/Average-case Equivalence*, In Proc. of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, pp. 284-293, 1997.
2. O. Goldreich, S. Goldwasser and S. Halevi, *Public-key Cryptosystems from Lattice Reduction Problems*, In Proc. of Crypto'97, volume 1294 of LNCS, pp. 112-131, Springer-Verlag, 1997.
3. P. Nguyen, *Cryptanalysis for the Goldreich-Goldwasser-Halevi Cryptosystem form Crypto'97*, In Proc. of Crypto'99, volume 1666 of LNCS, pp. 288-304, Springer-Verlag, 1999.

4. P. van Emde Boas, *Another NP-complete Problem and the Complexity of Computing Short Vectors in a Lattice*, Technical Report 81-04, Mathematische Instituut, University of Amsterdam, 1981.
5. I. Dinur, G. Kindler and S. Safra, *Approximating-CVP to within Almost-Polynomial Factors is NP-hard*, Electronic Colloquium on Computational Complexity, Report No. 48, 1998.
6. D. Micciancio, *On the hardness of the Shortest Vector Problem*, Ph. D. Dissertation, Massachusetts Institute of Technology, Sept. 1998.
7. C. P. Schnorr, *A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithm*, In Theoretical Computer Science, volume 53, pp. 201-224, 1987.

Secure Length-Saving ElGamal Encryption under the Computational Diffie-Hellman Assumption

Joonsang Baek, Byoungcheon Lee, and Kwangjo Kim

Information and Communications University(ICU), 58-4 Hwaam-dong,
Yusong-gu, Taejon, 305-348, S. Korea
{mohi, sultan, kkj}@icu.ac.kr

Abstract. A design of secure and efficient public key encryption schemes under weaker computational assumptions has been regarded as an important and challenging task. As far as the ElGamal-type encryption is concerned, some variants of the original ElGamal encryption scheme whose security depends on weaker computational assumption have been proposed: Although the security of the original ElGamal encryption is based on the decisional Diffie-Hellman assumption (DDH-A), the security of a recent scheme, such as Pointcheval's ElGamal encryption variant, is based on the weaker assumption, the computational Diffie-Hellman assumption (CDH-A). In this paper, we propose a length-saving ElGamal encryption variant whose security is based on CDH-A and analyze its security in the random oracle model. The proposed scheme is length-efficient which provides a shorter ciphertext than that of Pointcheval's scheme and provably secure against the chosen-ciphertext attack.

1 Introduction

1.1 Encryption Schemes Based on Diffie-Hellman Assumption

Ever since Diffie and Hellman [9] originally proposed the concept of public-key cryptosystem, extensive research has been performed in this field. In particular, the public-key encryption scheme proposed by ElGamal [10] has attracted considerable attention. When ElGamal proposed his public-key encryption scheme, it was widely believed that the security of this scheme is based on the computational assumption called "Diffie-Hellman assumption". Roughly speaking, the Diffie-Hellman assumption says that for a cyclic group G , an adversary who sees g^x and g^y cannot efficiently compute g^{xy} . Often, G is defined as a multiplicative group of a large prime modulo p , *i.e.*, $\mathbb{Z}_p^*$ where g is a generator and $x, y \in \mathbb{Z}_q$. Note here that q is a large prime such that $q|p-1$.

It may be true that the security of ElGamal encryption scheme depends on the Diffie-Hellman assumption since an adversary attacking this scheme cannot decrypt a ciphertext (g^y, mg^{xy}) of a message m without computing g^{xy} . However, indistinguishability [12], which has been accepted as a general security notion of encryption schemes, does not require an attacker to decrypt the whole

message. In the notion of indistinguishability, security of encryption scheme implies that an adversary cannot distinguish ciphertexts of two messages chosen by him-or herself. Consequently, it seems that the security of ElGamal encryption should depend on some stronger assumption rather than the Diffie-Hellman assumption. In fact, Tsionis and Yung [14] showed that the security of ElGamal encryption scheme is not based on the Diffie-Hellman assumption but based on the stronger Decisional Diffie-Hellman assumption (DDH-A). DDH-A says that an adversary who sees two distributions (g^x, g^y, g^{xy}) and (g^x, g^y, R) , where R is a randomly chosen-string whose length is the same as g^{xy} , cannot distinguish these two distributions. Hence the Diffie-Hellman assumption is often called the computational Diffie-Hellman assumption (CDH-A) for the purpose of emphasizing an adversary's inability to compute the Diffie-Hellman key, g^{xy} . Throughout this paper, we will use the term CDH-A to refer to the Diffie-Hellman assumption.

1.2 Chosen Ciphertext Security

Ever since Zheng and Seberry [15] initiated a full-scale research on adaptive chosen-ciphertext attacks, the design of public-key encryption schemes has trended toward the prevention of these attacks. In the adaptive chosen-ciphertext attack, an adversary is permitted to access a decryption function as well as an encryption function. The adversary may use this decryption function on ciphertexts chosen after obtaining the challenge ciphertext, with the only restriction that the adversary may not ask for the decryption of the challenge ciphertext itself.

Several security notions on the (adaptive or non-adaptive) chosen-ciphertext attack including non-malleability [8] were formalized and the relationship among them was shown in [3]. Public-key encryption schemes secure against the adaptive chosen-ciphertext attack proposed so far include OAEP [5] (based on the RSA function), the Cramer-Shoup scheme [7] (based on the DDH-A), DHAES [1] (based on the hash Diffie-Hellman assumption (HDH-A)), and the Fujisaki-Okamoto(F-O) scheme [11] (based on the security of any semantically secure public-key encryption schemes). More recently, a general method for converting any partially trapdoor one-way function to the public-key encryption scheme that is secure against the chosen-ciphertext attack was proposed by Pointcheval [13].

The Cramer-Shoup scheme is said to be unique since it does not impose any ideal assumption on the underlying hash function as other schemes do. Although the use of the ideal hash function model, *i.e.*, the random oracle model [4], is still controversial [6], this paradigm often yields much more efficient schemes than those in the standard model [2].

We note here that the underlying computational assumption of Cramer-Shoup scheme is DDH-A, which is much stronger than CDH-A, although the random oracle model is not used in this scheme. The situation remains the same in the ElGamal version of the F-O scheme. However, underlying computational assumption of the ElGamal version of recent Pointcheval's scheme is CDH-A, which is weaker than DDH-A. On the other hand, one deficiency of this scheme

is a message expansion. That is, to encrypt a message m , one must compute $(g^{H(m||s)}, rX^{H(m||s)}, G(r) \oplus (m||s))$, where $X(=g^x)$ is a public key, $r \in \mathbb{Z}_p^*$ and $s \in \mathbb{Z}_q$ are appropriate lengths of random strings. Here, both G and H are random oracles. Consequently, the lengths of a ciphertext is 1.5 times longer than that of the original ElGamal version of the F-O scheme.

Based on aforementioned discussions, we propose another ElGamal encryption variant provably secure against chosen-ciphertext attack in the random oracle model. The underlying computational assumption of the proposed scheme is based on CDH-A, but the length of ciphertext is reduced compared with the Pointcheval's scheme.

The organization of this paper is as follows: We briefly review the notions of chosen-ciphertext security for public-key encryption schemes in Section 2. In Section 3, we describe the proposed scheme and analyze its security. In Section 4, comparison of the proposed scheme with other ElGamal variants is provided and concluding remarks will follow in the final section.

2 Notions of Security

Although there are several security notions on the chosen-plaintext attacks and the chosen-ciphertext attacks, we briefly review two notions, the indistinguishability-chosen plaintext attack (IND-CPA) [3,12] and the plaintext awareness (PA) [3,5].

Security against the chosen-plaintext attack for public-key encryption schemes is defined by using the following experiment: Let $\mathcal{A}$ be an adversary with two algorithms A_1 and A_2 . The “find”-stage algorithm A_1 is run on the public key, pk . At the end of A_1 's execution, it outputs a 3-tuple (m_0, m_1, s) where m_0 and m_1 are messages that have the same length and s is a state information. Then one of m_0 and m_1 is selected at random and ciphertext y is determined by encrypting m_b ($b \in_R \{0, 1\}$) under pk . The job of the “guess”-stage algorithm A_2 is to determine if y was selected as the encryption of m_0 or m_1 , namely to determine the bit b . If the probability that A_2 outputs b is negligible, we say that the public-key encryption scheme is secure in the sense of IND-CPA. Now, we formally define this experiment as follows:

Definition 1 (IND-CPA). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, where $\mathcal{K}$, $\mathcal{E}$, and $\mathcal{D}$ denote a key generation algorithm, an encryption algorithm, and a decryption algorithm, respectively. Let $\mathcal{A}(A_1, A_2)$ be an adversary where A_1 denotes a “find”-stage algorithm and A_2 denotes a “guess”-stage algorithm. Also, let (sk, pk) be a secret and public key pair and let s be a state information. If the advantage of $\mathcal{A}$

$$\text{Adv}_{\mathcal{A}, \Pi}^{\text{IND-CPA}} = 2 \cdot [(sk, pk) \leftarrow \mathcal{K}; (m_0, m_1, s) \leftarrow A_1(\text{find}, s); b \leftarrow \{0, 1\}; y \leftarrow \mathcal{E}_{pk}(m_b) : A_2(\text{guess}, pk, s, y) = b] - 1$$

is negligible, we say that Π is secure in the sense of IND-CPA.

The plaintext awareness (PA), first defined by Bellare and Rogaway [5], formalizes an adversary's inability to create the ciphertext y without "knowing" its corresponding plaintext x .

We note that PA has only been defined in the random oracle model. An adversary $\mathcal{B}$ for PA is given a public key pk and access to the random oracle H . We also provide $\mathcal{B}$ with an oracle for $\mathcal{E}_{pk}^H$. The adversary outputs a ciphertext y . To be PA, the adversary $\mathcal{B}$ should necessarily know the decryption m of its output. To formalize this, it is required that there exists an algorithm K (knowledge extractor) that could have output m just by looking at the public key, $\mathcal{B}$'s H -queries and their answers, and the answers to $\mathcal{B}$'s queries to $\mathcal{E}_{pk}^H$. The following is a formal definition of PA.

Definition 2 (PA). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, let $\mathcal{B}$ be an adversary, let $hH = \{(h_1, H_1), (h_2, H_2), \dots, (h_{q_H}, H_{q_H})\}$ be a list of all of $\mathcal{B}$'s oracle queries, $h_1, h_2, \dots, h_{q_H}$, and the corresponding answers $H_1, H_2, \dots, H_{q_H}$, and let K be a knowledge extractor. Let $C = \{y_1, y_2, \dots, y_{q_H}\}$ denote the answers(ciphertexts) as a result of $\mathcal{E}_{pk}^H$ -queries. For any $k \in \mathbb{N}$ define

$$\text{Succ}_{K, \mathcal{B}, \Pi}^{\text{PA}} = \Pr[H \leftarrow \text{Hash}; (pk, sk) \leftarrow \mathcal{K}; (hH, C, y) \leftarrow \text{run}\mathcal{B}^{H, \mathcal{E}_{pk}^H}(pk) : \\ K(hH, C, y, pk) = D_{sk}^H(y)].$$

For $y \notin C$, we say that K is a $\lambda(k)$ -extractor if K has running time polynomial in the length of its inputs and for every $\mathcal{B}$, $\text{Succ}_{K, \mathcal{B}, \Pi}^{\text{PA}} \geq \lambda(k)$. We say that Π is secure in the sense of PA if Π is secure in the sense of IND-CPA and there exists a $\lambda(k)$ -extractor K where $1 - \lambda(k)$ is negligible.

3 Secure Length-Saving ElGamal Encryption Variant

3.1 Description of the Proposed Scheme

Our motivation for constructing the public key encryption scheme whose security relies on CDH-A is to apply random oracle G to Diffie-Hellman key g^{xy} . Since G is assumed to be a random oracle, $G(g^{xy})$ does not reveal any (partial) information about g^{xy} . Hence, to gain any advantage, the adversary must compute g^{xy} . Also, to provide PA, we apply another random oracle H to message m concatenated by some random string s . This motivation leads to the proofs for the theorems provided later in this section. A concrete description of the proposed scheme Π is as follows:

Secure Length-saving ElGamal Encryption Variant $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$

- Key generator $\mathcal{K}$
 $(pk, sk) \leftarrow \mathcal{K}(1^k)$, $pk = (p, q, g, X(= g^x))$ and $sk = (p, q, g, x)$ where $x \in_R \mathbb{Z}_q$, $|p| = k = k_0 + k_1$, and $q|p - 1$, a large prime number.

- Hash Function (two random oracles)
 $H : \{0, 1\}^k \rightarrow \mathbb{Z}_q$, and $G : \mathbb{Z}_p^* \rightarrow \{0, 1\}^k$
- Encryption $\mathcal{E}$
 $\mathcal{E}_{pk}(m, s) = (\alpha, \beta) = (g^{H(m||s)} \bmod p, G(X^{H(m||s)} \bmod p) \oplus (m||s))$, where message $m \in \{0, 1\}^{k_0}$ and $s \leftarrow_R \{0, 1\}^{k_1}$
- Decryption $\mathcal{D}$

$$\mathcal{D}_{sk}(\alpha, \beta) = \begin{cases} [\beta \oplus G(\alpha^x \bmod p)]^{k_0} & \text{if } \alpha = g^{H(\beta \oplus G(\alpha^x \bmod p))} \\ \varepsilon(\text{null}) & \text{otherwise} \end{cases}$$

where $[\beta \oplus G(\alpha^x \bmod p)]^{k_0}$ denotes the first k_0 bit of $[\beta \oplus G(\alpha^x \bmod p)]$.

3.2 Security Analysis

In this section, we show that our ElGamal encryption variant is secure in the sense of IND-CPA under CDH-A and there exists a knowledge extractor K .

Note that the security in the sense of IND-CPA and the existence of a knowledge extractor mean the security in the sense of PA. By the result of [3], this implies security against the adaptive chosen-ciphertext attack (IND-CCA2)

Theorem 1. *If there exists an adversary attacking the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in a chosen-plaintext scenario, then we can construct an adversary that breaks CDH-A in the random oracle model with non-negligible probability.*

Proof. Let $\mathcal{A} = (A_1, A_2)$ be an adversary attacking $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in a chosen-plaintext scenario and ϵ be an advantage of $\mathcal{A}$. Recall that A_1 denotes the “find”-stage algorithm and A_2 denotes the “guess”-stage algorithm. Assume that both G and H are random oracles. Our proving strategy is to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks CDH-A. Suppose that $X(= g^x)$ and $Y(= g^y)$ are given to $\mathcal{B}$. $\mathcal{B}$ works as follows:

- First give X , as a public key, to $\mathcal{A}$ and run $\mathcal{A}$. When A_1 makes any oracle query j to G , $\mathcal{B}$ chooses a random string in $\{0, 1\}^k$ and answers it as $G(j)$. Similarly, if A_1 makes any oracle query j to H , $\mathcal{B}$ chooses a random string in $\mathbb{Z}_q$ and answers it as $H(j)$. A_1 finally outputs two messages m_0 and m_1 . $\mathcal{B}$ then selects $b \in \{0, 1\}$ at random, takes a random string T in $\{0, 1\}^k$ for $G(X^y)$, and outputs $(\alpha, \beta) = (Y, T \oplus (m_b||s))$ as a ciphertext. Then it defines $H(m_b||s)$ as y and $G(X^y)$ as $\beta \oplus (m_b||s)$.
- The ciphertext (α, β) is provided as an input to A_2 . Then, A_2 makes oracle queries and $\mathcal{B}$ answers as above.
- When A_2 returns its answer $d \in \{0, 1\}$, $\mathcal{B}$ returns a set of all the oracle queries asked to G during the attack.

Now let us define the following two events, $AskG$ and $AskH$.

- $AskG$: The query X^y was made to G .

- $AskH$: The query $m||s$ for some messages m and s chosen at the beginning by $\mathcal{B}$, is made to H .

If $Succ\mathcal{A}$ is an event that A_2 correctly guesses the bit b , *i.e.*, outputs $d = b$, then the advantage of the adversary is defined by $2 \cdot \Pr[Succ\mathcal{A}] - 1$. But, if X^y or $(m_b||s)$ has been already asked to G or H , respectively, the attacker succeeds in guessing the bit b . Hence, $2 \cdot \Pr[Succ\mathcal{A}] - 1 \geq \epsilon$. Since the adversary gains no advantage without $AskG$ or $AskH$, we obtain

$$\Pr[Succ\mathcal{A}] \leq \frac{1}{2} + \frac{\Pr[AskG \vee AskH]}{2}$$

and this leads to $\Pr[AskG \vee AskH] \geq \epsilon$.

Furthermore,

$$\begin{aligned} \Pr[AskG \vee AskH] &= \Pr[AskG] + \Pr[AskH \wedge \neg AskG] \\ &= \Pr[AskG] + \Pr[AskH|\neg AskG]\Pr[\neg AskG] \\ &\leq \Pr[AskG] + \Pr[AskH|\neg AskG] \end{aligned}$$

Yet, the probability that the event $AskH$ takes place is very small provided that $\neg AskG$ is true. More precisely,

$$\Pr[AskH|\neg AskG] \leq \frac{q_H}{2^{k_1}}.$$

Therefore, we have

$$\Pr[AskG] \geq \epsilon - \frac{q_H}{2^{k_1}}.$$

This implies that the probability that X^y lies in the set of all the oracle queries to G is greater than $\epsilon - \frac{q_H}{2^{k_1}}$. Hence if the advantage ϵ of $\mathcal{A}$ is non-negligible, $\mathcal{B}$ breaks CDH-A with non-negligible probability. $\square$

Now we construct a knowledge extractor K . Note that the existence of K implies security in the sense of PA under the assumption that Π is secure in the sense of IND-CPA.

Theorem 2. *Let $\mathcal{B}$ be an adversary for PA. Then there exists a knowledge $\lambda(k)$ -extractor K and hence $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is secure in the sense of PA.*

Proof. Since we have shown that Π is secure in the sense of IND-CPA, we only need to construct a knowledge-extractor K . Assume that $gG = \{(g_1, G_1), (g_2, G_2), \dots, (g_{q_G}, G_{q_G})\}$, $hH = \{(h_1, H_1), (h_2, H_2), \dots, (h_{q_H}, H_{q_H})\}$ (all the random oracle query-answer pairs of $\mathcal{B}$), $C = \{y_1, y_2, \dots, y_E\}$ (a set of ciphertexts that $\mathcal{B}$ has obtained from the interaction with the random oracles and the encryption oracle), $y = (\alpha, \beta) \notin C$ (a ciphertext produced by $\mathcal{B}$ which is not in C), and the public key X are given to K . The knowledge extractor K works as follows:

- For all gG and hH , K checks that there exists a pair (g_q, h_r) such that $y = (\alpha, \beta) = (g^{H_r}, G_q \oplus h_r)$.
- If there exists one pair, K returns $m = [h_r]^{k_0}$ and s . Otherwise, outputs $\varepsilon(\text{null})$.

Next we think of the probability that K outputs the plaintext m correctly, namely $m = \mathcal{D}_{sk}(y)$. Let $Fail$ be an event that $m \neq \mathcal{D}_{sk}(y)$ and let $AskG$ be an event that there exists a pair (g_q, G_q) in the list gG such that $y = (\alpha, \beta) = (g^{H_r}, G_q \oplus h_r)$ for some (h_r, H_r) in the list hH . Similarly, let $AskH$ be an event that there exists a pair (h_r, H_r) in the list hH such that $y = (\alpha, \beta) = (g^{H_r}, G_q \oplus h_r)$ for some (g_q, G_q) in the list gG . Then,

$$\begin{aligned} \Pr[Fail] &= \Pr[Fail|AskG \wedge AskH]\Pr[AskG \wedge AskH] + \\ &\quad \Pr[Fail|\neg AskG \vee \neg AskH]\Pr[\neg AskG \vee \neg AskH] \\ &\leq 0 + \Pr[Fail|\neg AskG \vee \neg AskH]. \end{aligned}$$

We now determine the upper bound of $\Pr[Fail|\neg AskG \vee \neg AskH]$. For *valid* y , there exist h and g such that $y = (g^{H(h)}, G(g) \oplus h)$. As $y \notin C$, it follows that $h \neq \mathcal{D}(y_i)$ for every $y_i \in C$. However,

$$\begin{aligned} \Pr[valid|\neg AskG \vee \neg AskH] &= \frac{\Pr[valid \wedge (\neg AskG \vee \neg AskH)]}{\Pr[(\neg AskG \vee \neg AskH)]} \\ &\leq \frac{\Pr[valid \wedge \neg AskH]}{\Pr[\neg AskH]} + \frac{\Pr[valid \wedge \neg AskG \wedge \neg AskH]}{\Pr[\neg AskG]} \\ &\leq \Pr[valid|\neg AskH] + \Pr[valid|\neg AskG] \\ &\leq \frac{1}{2^k} + \frac{1}{2^{k_1}}. \end{aligned}$$

On the other hand, if $\neg AskG$ or $\neg AskH$ is true, from the construction of K , it always outputs $\varepsilon(\text{null})$, i.e., y is *invalid*. This means that $\Pr[Fail|\neg AskG \vee \neg AskH] = \Pr[valid|\neg AskG \vee \neg AskH] \leq \frac{1}{2^k} + \frac{1}{2^{k_1}}$. Namely, the probability of rejection of valid ciphertext is upper-bounded by $1/2^k + 1/2^{k_1}$. Consequently,

$$\lambda(k) = 1 - \Pr[Fail] \geq 1 - \frac{1}{2^k} - \frac{1}{2^{k_1}}.$$

□

As mentioned before, we get the following corollary from Theorems 1 and 2.

Corollary 1. *The proposed scheme is secure in the sense of IND-CCA2.*

4 Comparison with Other Schemes

We compare the length of ciphertext of the proposed scheme with the original ElGamal encryption scheme and other ElGamal-type encryption schemes such as ElGamal encryption variant of the F-O scheme, and the Pointcheval's ElGamal encryption variant.

For comparison, we briefly describe how four schemes encrypt a message m .

- ElGamal scheme : $(g^y, X^y m)$
- F-O scheme : $(g^{H(m||s)}, X^{H(m||s)} \oplus (m||s))$
- Pointcheval's scheme : $(g^{H(m||s)}, X^{H(m||s)} r, G(r) \oplus (m||s))$
- The proposed scheme : $(g^{H(m||s)}, G(X^{H(m||s)}) \oplus (m||s))$

We summarize the cryptographic characteristics of four schemes in Table 1.

Table 1. Comparison with Other ElGamal Variants, where: $k = |Z_p^*|$, RO = Random Oracle, E= Exponentiation, H= Random oracle computation, Comp. for Enc.= Computation for Encryption, Comp. for Dec.=Computation for Decryption

	ElGamal	F-O	Pointcheval	Proposed scheme
Length	$2k$	$2k$	$3k$	$2k$
Number of ROs	None	1	2	2
Assumption	DDH-A	DDH-A	CDH-A	CDH-A
Security	IND-CPA	IND-CCA2	IND-CCA2	IND-CCA2
Comp. for Enc.	2E	2E+H	2E+2H	2E+2H
Comp. for Dec.	E	2E+H	2E+2H	2E+2H

As can be seen from the table, the proposed scheme guarantees sound security and length-efficiency. Under the CDH-A, it is secure in the sense of IND-CCA2. We now provide a more detailed explanation on the length of a ciphertext. In the F-O scheme, the length of a ciphertext is $2k$ where $k = |Z_p^*|$. A ciphertext of the proposed scheme has the same length as those of the original ElGamal scheme and the F-O scheme, when the length of output of G , which is used as the random oracle, is set to k . In the Pointcheval's scheme, the length of ciphertext is extended to $3k$. Compared with the Pointcheval's scheme, the proposed scheme effectively reduces the length of a ciphertext under the same circumstances, *i.e.*, the security of both schemes is based on CDH-A and two random oracles are used. Note that the message to ciphertext ratio of the original ElGamal scheme is the largest since no additional random string follows the message m being encrypted. However, as widely known, the original ElGamal scheme is insecure against chosen-ciphertext attack. The message to ciphertext ratios of other three schemes are the same.

As also can be seen from the table, the computation cost required in the proposed scheme to encrypt and decrypt messages is estimated to be the same as that of the Pointcheval's scheme. Note that we have omitted the computation required to generate public key.

Finally, we mention about implementation of the random oracle G . To implement this function, one can use the heuristic method described in [4] and [5] as follows:

$$G(X^y) = g(\langle 0 \rangle, X^y) || g(\langle 1 \rangle, X^y) || g(\langle 2 \rangle, X^y) || \dots,$$

where g is an efficient cryptographic hash function such as SHA-1 or MD5 which outputs 160 bits or 128 bits, respectively, and the notation $\langle i \rangle$ denotes a binary 32-bit word encoding of integer i .

5 Concluding Remarks

In this paper we have proposed another variant of the ElGamal encryption scheme. The security of the proposed scheme depends on CDH-A, which is much weaker than DDH-A. Moreover, the length of a ciphertext is reduced compared with the recent Pointcheval's ElGamal variant, which is based on CDH-A. Also, the proposed scheme provides the same degree of computational efficiency as other proposed schemes.

However, as done in other practical schemes, the random oracle model is employed to provide provable security. A construction of "practical" public-key encryption schemes secure against active adversaries without random oracle other than the one in [7] is an interesting and meaningful future work.

Acknowledgements. The authors would like to thank the anonymous referees for their helpful comments and Prof. Ben Lee at ICU for his proof reading of the final version of this paper.

References

1. M. Abdalla, M. Bellare, and P. Rogaway, "DHAES: An Encryption Scheme Based on Diffie-Hellman Problem", *IEEE P1363a Submission*, 1998, Available at <http://grouper.ieee.org/groups/1363/addendum.html>.
2. M. Bellare, "Practice-Oriented Provable-Security", *In the First International Workshop on Information Security - Proceedings of ISW'97, LNCS 1396*, Springer-Verlag, 1998.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes", *In Advances in Cryptology - Proceedings of Crypto'98, LNCS 1462*, pp.26-45, Springer-Verlag, 1998.
4. M. Bellare and P. Rogaway, "Random Oracles are Practical : A Paradigm for Designing Efficient Protocols", *ACM Conference on Computer and Communications Security*, pp.62-73, 1993.
5. M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption - How to Encrypt with RSA", *In Advances in Cryptology - Proceedings of Eurocrypt'94, LNCS 950*, pp.92-111, Springer-Verlag, 1995.
6. R. Canetti, O. Goldreich, and S. Halevi, "The Random Oracle Methodology, Revisited", *Proceedings of the 30th Annual Symposium on the Theory of Computing, ACM*, 1998.
7. R. Cramer and V. Shoup, "A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack", *In Advances in Cryptology - Proceedings of Crypto'98, LNCS 1462*, pp. 13-25, Springer-Verlag, 1998.
8. D. Dolev, C. Dwork, and M. Naor, "Non-Malleable Cryptography", *Proceedings of 23rd STOC.*, ACM Press 1991.

9. W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22(6), pp.644-654, 1976.
10. T. ElGamal, "A Public Key Cryptosystems and a Signature Schemes Based on Discrete Logarithms", *IEEE Transactions on Information Theory*, IT-31(4), pp.469-472, 1985.
11. E. Fujisaki and T. Okamoto, "How to Enhance the Security of Public-Key Encryption at Minimum Cost", *PKC'99, LNCS 1560*, pp.53-68, Springer-Verlag, 1999.
12. S. Goldwasser and S. Micali, "A Probabilistic Encryption", *Journal of Computer and System Sciences*, 28, pp. 270-299, 1984.
13. D. Pointcheval, "Chosen-Ciphertext Security for any One-Way Cryptosystem", *PKC'2000, LNCS 1751*, pp.129-146, Springer-Verlag, 2000.
14. Y. Tsionis and M. Yung, "On the Security of ElGamal Based Encryption", *PKC'98, LNCS 1431*, pp.117-134, Springer-Verlag, 1998.
15. Y. Zheng and J. Seberry. "Practical Approaches to Attaining Security Against Adaptively Chosen Ciphertext Attacks", *In Advances in Cryptology - Proceedings of Crypto'92, LNCS 740*, pp. 292-304, Springer-Verlag, 1993.

Efficient Scalar Multiplications on Elliptic Curves without Repeated Doublings and Their Practical Performance

Yasuyuki Sakai¹ and Kouichi Sakurai²

¹ Mitsubishi Electric Corporation,
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan
`ysakai@iss.isl.melco.co.jp`

² Kyushu University,
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-81, Japan
`sakurai@csce.kyushu-u.ac.jp`

Abstract. We introduce efficient algorithms for scalar multiplication on elliptic curves defined over $\mathbb{F}_p$. The algorithms compute $2^k P$ directly from P , where P is a random point on an elliptic curve, without computing the intermediate points, which is faster than k repeated doublings. Moreover, we apply the algorithms to scalar multiplication on elliptic curves, and analyze their computational complexity. As a result of their implementation with respect to affine (resp. weighted projective) coordinates, we achieved an increased performance factor of 1.45 (45%) (resp. 1.15 (15%)) in the scalar multiplication of the elliptic curve of size 160-bit.

Keywords. Elliptic Curve Cryptosystems, Scalar Multiplication, Window Method, Coordinate System, Implementation

1 Introduction

Elliptic curve cryptosystems, which were suggested by Miller [M185] and Koblitz [K087], are now widely used in various security services. IEEE and other standards bodies such as ANSI and ISO are in the process of standardizing elliptic curve cryptosystems. Therefore, it is very attractive to provide algorithms that allow efficient implementation [CMO98, GP97, KT92, MOC97a, MOC97b, So97]D

Encryption/decryption or signature generation/verification schemes require computation of scalar multiplication. The computational performance of cryptographic protocols with elliptic curves strongly depends on the efficiency of scalar multiplication. Thus, fast scalar multiplication is essential for elliptic curve cryptosystems. In typical methods for scalar multiplication, an addition of two points and a doubling of a point, are calculated repeatedly, but the point doublings are quite costly. There are several ways to speed-up scalar multiplication, such as: (1) reducing the number of additions, (2) speeding-up doubling, and (3) using a mixed coordinate strategy [CMO98]. Our contribution will deal with the second approach.

One method to increase the speed of doublings is *direct* computation of several doublings, which computes $2^k P$ directly from $P \in E(\mathbb{F}_q)$, without computing

the intermediate points $2P, 2^2P, \dots, 2^{k-1}P$. The concept of direct computation was first suggested by Guajardo and Paar in [GP97]. They formulated algorithms for direct computation of $4P, 8P$ and $16P$ on elliptic curves over $\mathbb{F}_{2^n}$ in terms of affine coordinates. Recent related results include a formula for computing $4P$ on elliptic curves over $\mathbb{F}_p$ in affine coordinates by Müller [Mu97] and a formula for computing $4P$ on elliptic curves over $\mathbb{F}_p$ in projective coordinates by Miyaji, Ono and Cohen [MOC97a]. These formulae are more efficient than repeated doublings. However, the known formulae work only with small k (2, 3 or 4) and formulae in terms of weighted projective coordinates have not been given. One remarkable result was given by Cohen, Miyaji and Ono [CMO98]. They used a redundant representation of points such as (X, Y, Z, aZ^4) . With this representation, point doubling can be accomplished with complexity $4S + 4M$, where S and M denote a squaring and a multiplication in $\mathbb{F}_p$ respectively. Itoh et al. also gave a similar method for doubling [ITTTK98]. This representation is called *modified jacobian coordinates* [CMO98]. We can use this coordinate system for direct computation of several doublings. However, addition of points in modified jacobian coordinates is relatively costly compared to weighted projective coordinates. Therefore, one possible strategy for efficient scalar multiplication is to mix several coordinate systems (See [CMO98]).

In this paper, we propose efficient algorithms for speeding-up elliptic curve cryptosystems with curves over $\mathbb{F}_p$ in terms of affine coordinates and weighted projective coordinates. We construct efficient formulae which compute 2^kP directly for $\forall k \geq 1$. Our formulae compute 2^kP directly from $P \in E(\mathbb{F}_p)$ without computing the intermediate points $2P, 2^2P, \dots, 2^{k-1}P$. In the case of affine coordinates, our formula has computational complexity $(4k+1)S + (4k+2)M + I$, where I denotes an inversion in $\mathbb{F}_p$. This is more efficient than k repeated doublings, which requires k inversions. When implementing our direct computation method, experimental results show that computing $16P$ achieves a 90 percent performance increase over 4 doublings in affine coordinates.

Moreover, we show a method of elliptic scalar multiplication that is combined with our direct computation methods. This method is based on a sliding signed binary window method [KT92]. We also implement scalar multiplication and discuss the efficiency. Our implementation results show that in the case of affine (resp. weighted projective) coordinates, we achieved a 45scalar multiplication of the elliptic curve of size 160-bit. Moreover, our implementation with the new method shows that when $\log_2 p$ is relatively large (384 or larger), scalar multiplication in affine coordinates can be faster than in weighted projective coordinates.

The algorithms proposed in this paper do not depend on specific curve parameters. Therefore, our methods can be applied to any elliptic curve defined over $\mathbb{F}_p$.

2 Previous Work

In this section, we summarize known algorithms for point doubling and direct computation of several doublings.

Throughout this paper, we will use the following notations. Let $\mathbb{F}_p$ denote a prime finite field with p elements. We consider an elliptic curve E given by

$$E : Y^2 = X^3 + aX + b \quad (a, b \in \mathbb{F}_p, p > 3, 4a^3 + 27b^2 \neq 0)$$

Let $P_1 = (x_1, y_1)$ and $P_{2^k} = 2^k P_1 = (x_{2^k}, y_{2^k}) \in E(\mathbb{F}_p)$. Let S , M and I denote a squaring, a multiplication and an inversion in $\mathbb{F}_p$, respectively. When we estimate computational efficiency, we will ignore the cost of a field addition, as well as the cost of multiplication by small constants.

2.1 Point Doubling

In terms of affine coordinates, point doubling can be accomplished as follows: Assume $P_1 = (x_1, y_1) \neq \mathcal{O}$, where $\mathcal{O}$ denotes the point at infinity. The point $P_2 = (x_2, y_2) = 2P_1$ can be computed as follows:

$$\begin{aligned} x_2 &= \lambda^2 - 2x_1 \\ y_2 &= (x_1 - x_2)\lambda - y_1 \\ \lambda &= \frac{3x_1^2 + a}{2y_1} \end{aligned} \tag{1}$$

The formulae above have computational complexity $2S + 2M + I$ [IEEE].

In terms of weighted projective coordinates, doubling can be accomplished as follows. Assume $P_1 = (X_1, Y_1, Z_1) \neq \mathcal{O}$, The point $P_2 = (X_2, Y_2, Z_2) = 2P_1$ can be computed as follows.

$$\begin{aligned} X_2 &= M^2 - 2S \\ Y_2 &= M(S - X_2) - T \\ Z_2 &= 2Y_1 Z_1 \\ M &= 3X_1^2 + aZ_1^4 \\ S &= 4X_1 Y_1^2 \\ T &= 8Y_1^4 \end{aligned} \tag{2}$$

In cases of general curves, the formulae above have computational complexity $6S + 4M$ [IEEE].

2.2 Direct Doubling

The concept of using direct computation of $2^k P$ for efficient elliptic scalar multiplication was first proposed by Guajardo and Paar in [GP97]. They formulated algorithms for computing $4P$, $8P$ or $16P$ on elliptic curves over $\mathbb{F}_{2^n}$ in terms of affine coordinates. In recent years, several authors have written methods to compute $2^k P$ directly (but limited to small k). The following section summarizes previous work on direct computation of several doublings.

1. Guajardo and Paar [GP97] proposed formulae for computing $4P$, $8P$ and $16P$ on elliptic curves over $\mathbb{F}_{2^n}$ in terms of affine coordinates.

2. Müller [MU97] proposed a formula for computing $4P$ on elliptic curves over $\mathbb{F}_p$ in terms of affine coordinates.
3. Miyaji, Ono and Cohen [MOC97a] proposed a formula for computing $4P$ on elliptic curves over $\mathbb{F}_p$ in terms of projective coordinates.
4. Han and Tan [HT99] proposed formulae for computing $3P$, $5P$, $6P$ and $7P$, etc, on elliptic curves over $\mathbb{F}_{2^n}$ in terms of affine coordinates.

The formulae above can efficiently compute $2^k P$ compared to computing k doublings. However, the formulae listed below have not been explicitly given.

1. General (i.e., $\forall k \geq 1$) formulae for direct computations in elliptic curves over $\mathbb{F}_p$ in terms of affine coordinates.
2. Formulae for direct computations in terms of weighted projective coordinates.

In later sections, we will formulate the above algorithms, and analyze their computational complexity. In the case of $k = 2$, i.e., $4P$, on elliptic curves over $\mathbb{F}_p$ in affine coordinates, our algorithms are more efficient than Müller's algorithm.

We should remark that the algorithm proposed by Cohen, Miyaji and Ono in [CMO98] can be efficiently used for direct computation of several doublings. The authors call their algorithm a “*modified jacobian*” coordinate system. The coordinate system uses (redundant) mixed representation, e.g., (X, Y, Z, aZ^4) . Doubling in terms of the modified jacobian coordinates has computational advantages over weighted projective (jacobian) coordinates.

3 Direct Computations of $2^k P$ in Affine Coordinates

In this section, we provide formulae for direct computation of $2^k P$, where $\forall k \geq 1$, in terms of affine coordinates. In the next section, we will show formulae in terms of weighted projective coordinates and discuss their computational efficiency.

In the case of affine coordinates, direct computation of several doublings may be significantly more efficient, as suggested in [GP97], because we can construct formulae that require only one modular inversion, as opposed to the k inversions that k separate doubling operations would require for computing $2^k P$.

Modular inversion is generally more expensive than modular multiplication [WMPW98]. Therefore, direct computation of several doublings may be effective in elliptic scalar multiplication in terms of affine coordinates.

3.1 Doubling

We begin by showing doubling formulae with the purpose of constructing formulae for general (i.e., $\forall k \geq 1$) cases.

Let

$$\begin{aligned} A_1 &= x_1 \\ B_1 &= 3x_1^2 + a \\ C_1 &= -y_1 \\ D_1 &= 12A_1C_1^2 - B_1^2 \end{aligned}$$

Then the doubled point $P_2 = (x_2, y_2)$ of $P_1 = (x_1, y_1)$ can be computed as follows.

$$\begin{aligned} x_2 &= \frac{B_1^2 - 8A_1C_1^2}{(2C_1)^2} \\ y_2 &= \frac{8C_1^4 - B_1D_1}{(2C_1)^3} \end{aligned} \quad (3)$$

Note that although the denominator of x_2 differs from that of y_2 , the formulae above require only one inversion if we multiply the numerator of x_2 by $2C_1$.

The formulae have computational complexity $4S + 6M + I$. On the other hand, the formulae (1) have complexity $6S + 4M + I$. Therefore, it is clear that the formulae given in this subsection are inefficient. (We show the above formulae only for the purpose of constructing formulae for $k > 1$, as stated previously.)

3.2 Computing $4P$

In affine coordinates, quadrupling a point can be accomplished by the following:

$$\begin{aligned} A_2 &= B_1^2 - 8A_1C_1^2 \\ B_2 &= 3A_2^2 + 16aC_1^4 \\ C_2 &= -8C_1^4 - B_1(A_2 - 4A_1C_1^2) \\ D_2 &= 12A_2C_2^2 - B_2^2 \\ x_4 &= \frac{B_2^2 - 8A_2C_2^2}{(4C_1C_2)^2} \\ y_4 &= \frac{8C_2^4 - B_2D_2}{(4C_1C_2)^3} \end{aligned} \quad (4)$$

where $P_4 = (x_4, y_4) = 4P_1 = 4(x_1, y_1)$.

The formulae have computational complexity $9S + 10M + I$. Müller's formula [Mu97] has complexity $7S + 14M + I$. Therefore, our formula, above, is clearly more efficient. Moreover, we will show in a later section that computing $4P$ by the formula given in this section has less complexity than computing two separate doublings by (1).

3.3 Computing $8P$

In affine coordinates, computing $P_8 = (x_8, y_8) = 8P_1 = 8(x_1, y_1)$ can be accomplished by the following:

$$\begin{aligned} A_3 &= B_2^2 - 8A_2C_2^2 \\ B_3 &= 3A_3^2 + 256aC_1^4C_2^4 \\ C_3 &= -8C_2^4 - B_2(A_3 - 4A_2C_2^2) \\ D_3 &= 12A_3C_3^2 - B_3^2 \\ x_8 &= \frac{B_3^2 - 8A_3C_3^2}{(8C_1C_2C_3)^2} \\ y_8 &= \frac{8C_3^4 - B_3D_3}{(8C_1C_2C_3)^3} \end{aligned} \quad (5)$$

These formulae have computational complexity $13S + 14M + I$. We will later show that computing $8P$ by the formulae given in this section has less complexity than computing three separate doublings by (1).

3.4 Computing $16P$

In affine coordinates, computing $P_{16} = (x_{16}, y_{16}) = 16P_1 = 16(x_1, y_1)$ can be accomplished by the following:

$$\begin{aligned}
 A_4 &= B_3^2 - 8A_3C_3^2 \\
 B_4 &= 3A_4^2 + 4096aC_1^4C_2^4C_3^4 \\
 C_4 &= -8C_3^4 - B_3(A_4 - 4A_3C_3^2) \\
 D_4 &= 12A_4C_4^2 - B_4^2 \\
 x_{16} &= \frac{B_4^2 - 8A_4C_4^2}{(16C_1C_2C_3C_4)^2} \\
 y_{16} &= \frac{8C_4^4 - B_4D_4}{(16C_1C_2C_3C_4)^3}
 \end{aligned} \tag{6}$$

The formulae have computational complexity $17S + 18M + I$. We will show in a later section that computing $16P$ by the formulae given in this section has less complexity than computing four separate doublings by (1).

3.5 The Formulae for Computing 2^kP in Affine Coordinates

From the formulae, which compute $4P$, $8P$ or $16P$, given in the previous subsections, we can easily obtain general formulae that allow direct doubling $P \mapsto 2^kP$, where $k \geq 1$. The figure shown below describes these formulae, and their computational complexity is given as Theorem 1.

Algorithm 1: Direct computation of 2^kP in affine coordinates, where $k \geq 1$ and $P \in E(\mathbb{F}_p)$.

INPUT: $P_1 = (x_1, y_1) \in E(\mathbb{F}_p)$
 OUTPUT: $P_{2^k} = 2^kP_1 = (x_{2^k}, y_{2^k}) \in E(\mathbb{F}_p)$

Step 1. Compute A_1 , B_1 and C_1

$$\begin{aligned}
 A_1 &= x_1 \\
 B_1 &= 3x_1^2 + a \\
 C_1 &= -y_1
 \end{aligned}$$

Step 2. For i from 2 to k compute A_i , B_i and C_i

$$\begin{aligned}
 A_i &= B_{i-1}^2 - 8A_{i-1}C_{i-1}^2 \\
 B_i &= 3A_i^2 + 16^{i-1}a\left(\prod_{j=1}^{i-1} C_j\right)^4 \\
 C_i &= -8C_{i-1}^4 - B_{i-1}(A_i - 4A_{i-1}C_{i-1}^2)
 \end{aligned}$$

Step 3. Compute D_k

$$D_k = 12A_kC_k^2 - B_k^2$$

Step 4. Compute x_{2^k} and y_{2^k}

$$x_{2^k} = \frac{B_k^2 - 8A_kC_k^2}{\left(k \prod_{i=1}^k C_i\right)^2}$$

$$y_{2^k} = \frac{8C_k^4 - B_kD_k}{\left(k \prod_{i=1}^k C_i\right)^3}$$

Theorem 1. *In terms of affine coordinates, there exists an algorithm that computes 2^kP in at most $4k + 1$ squarings, $4k + 2$ multiplications, and one inversion in $\mathbb{F}_p$ for any point $P \in E(\mathbb{F}_p)$.*

Proof It is easy to prove by induction on k . $\square$

It should be noted that the point P_1 has to be an element with an order larger than 2^k . This requirement ensures that 2^kP will never equal $\mathcal{O}$.

3.6 Complexity Comparison

In this subsection, we compare the computational complexity of a direct doubling of 2^kP given in the previous subsection and separate k repeated doublings. The complexity of a doubling is estimated based on the algorithm given in [IEEE]. Table 1 shows the number of squarings S , multiplications M , and inversions I in $\mathbb{F}_p$. We should point out that our method reduces inversions at the cost of increased multiplications. Therefore, the performance of the new formulae depends on the cost factor of one inversion relative to one multiplication. For this purpose we introduce, as in [GP97], the notation of a “break-even point”. It is possible to express the time that it takes to perform one inversion in terms of the equivalent number of multiplications needed per inversion. In this comparison, we assume that one squaring has complexity $S = 0.8M$. We also assume that the cost of field addition and multiplication by small constants can be ignored.

As we can see from Table 1, if a field inversion has complexity $I > 10M$, one quadrupling may be more efficient than two separate doublings. In cases that $\mathbb{F}_p$ has a size of 160-bit or larger, it is extremely likely that $I > 10M$ in many implementations (i.e., [CMO98, WMPW98]). Moreover, in cases of $k > 2$, our direct computation method may be more efficient than individual doublings in most implementations.

4 Direct Computation of 2^kP in Weighted Projective Coordinates

In this section, we provide formulae for direct computation of 2^kP in terms of weighted projective coordinates and discuss the computational efficiency.

Table 1. Complexity comparison

Calculation	Method	Complexity			Break-Even Point
		S	M	I	
$4P$	Direct Doublings	9	10	1	$10M < I$
	Separate 2 Doublings	4	4	2	
$8P$	Direct Doublings	13	14	1	$7.2M < I$
	Separate 3 Doublings	6	6	3	
$16P$	Direct Doublings	17	18	1	$5.7M < I$
	Separate 4 Doublings	8	8	4	
$2^k P$	Direct Doublings	$4k + 1$	$4k + 2$	1	$\frac{3.6k+2.8}{k-1}M < I$
	Separate k Doublings	$2k$	$2k$	k	

4.1 General Formulae for Computing $2^k P$ in Weighted Projective Coordinates

In cases where field inversions are significantly more expensive than multiplications, it is preferable to use weighted projective coordinates (also referred to as *Jacobian coordinates*), where a triplet (X, Y, Z) corresponds to the affine coordinates $(X/Z^2, Y/Z^3)$ whenever $Z \neq 0$.

From Algorithm 1, we can immediately derive formulae for direct computation of $2^k P$ in terms of weighted projective coordinates. The computational complexity of Algorithm 2 is given as Theorem 2.

Algorithm 2: Direct computation of $2^k P$ in weighted projective coordinates, where $k \geq 1$ and $P \in E(\mathbb{F}_p)$.

INPUT: $P_1 = (X_1, Y_1, Z_1) \in E(\mathbb{F}_p)$

OUTPUT: $P_{2^k} = (X_{2^k}, Y_{2^k}, Z_{2^k}) = 2^k P_1 \in E(\mathbb{F}_p)$

Step 0. Mapping: $(X_1, Y_1, Z_1) \mapsto (X'_1, Y'_1, 1)$
 (if $Z_1 = 0$ terminate with $P_{2^k} = \mathcal{O}$)

Step 1. Compute A_1, B_1 and C_1

$$\begin{aligned} A_1 &= X'_1 \\ B_1 &= 3X'^2_1 + a \\ C_1 &= -Y'_1 \end{aligned}$$

Step 2. For i from 2 to k compute A_i, B_i and C_i

$$\begin{aligned} A_i &= B_{i-1}^2 - 8A_{i-1}C_{i-1}^2 \\ B_i &= 3A_i^2 + 16^{i-1}a\left(\prod_{j=1}^{i-1} C_j\right)^4 \\ C_i &= -8C_{i-1}^4 - B_{i-1}(A_i - 4A_{i-1}C_{i-1}^2) \end{aligned}$$

Step 3. Compute D_k

$$D_k = 12A_kC_k^2 - B_k^2$$

Step 4. Compute X_{2^k} , Y_{2^k} and Z_{2^k}

$$X_{2^k} = B_k^2 - 8A_kC_k^2$$

$$Y_{2^k} = 8C_k^4 - B_kD_k$$

$$Z_{2^k} = k \prod_{i=1}^k C_i$$

Theorem 2. *In terms of weighted projective coordinates, there exists an algorithm that computes $2^k P$ in at most $4k + 1$ squarings, $4k - 1$ multiplications in $\mathbb{F}_p$, except **Step 0**, for any point $P \in \mathbb{F}_p$. In particular, doublings can be computed with 4 squarings and 2 multiplications in $\mathbb{F}_p$.*

Proof It is easy to prove by induction on k . $\square$

Algorithm 2 does not require the following three multiplications, which Algorithm 1 (in affine coordinates) requires:

1. multiplication x by $k \prod_{i=1}^k C_i$
2. multiplication x by the inversion
3. multiplication y by the inversion

4.2 Quadrupling

In Algorithm 3, which follows, we describe formulae for computing $4P$ directly in terms of weighted projective coordinates. The algorithm does not require **Step 0** in Algorithm 2.

Algorithm 3: Computing $4P$ in weighted projective coordinates

INPUT: $P_1 = (X_1, Y_1, Z_1) \in E(\mathbb{F}_p)$
 OUTPUT: $P_4 = (X_4, Y_4, Z_4) = 4P_1 \in E(\mathbb{F}_p)$

Step 1. Compute A, B, C and D

$$\begin{aligned} A &= 3X_1^2 + aZ_1^4 \\ B &= A_j^2 - 8X_1Y_1^2 \\ C &= -8Y_1^4 + A_j(12X_1Y_1^2 - A^2) \\ D &= 16aY_1^4Z_1^4 + 3B^2 \end{aligned}$$

Step 2. Compute X_4, Y_4 and Z_4

$$\begin{aligned} X_4 &= -8BC^2 + D^2 \\ Y_4 &= -8C^4 + D(12BC^2 - D^2) \\ Z_4 &= 4Y_1Z_1C \end{aligned}$$

The algorithm above has computational complexity $10S + 9M$.

As we have stated before, if a modified jacobian representation (X, Y, Z, aZ^4) is used, a point doubling can be accomplished with complexity $4S + 4M$ [CMO98]. Therefore, a quadrupling can be accomplished with complexity $8S + 8M$. Itoh et al. also gave a way to quadruplicate with complexity $8S + 8M$ [ITTTK98]. However, in modified jacobian coordinates, the formulae for addition of points that have complexity $6S + 13M$ are more costly than those in weighted projective coordinates. One optimal way to speed-up scalar multiplication is to mix modified jacobian coordinates with projective, weighted projective, or Chudnovsky jacobian coordinates [CC86, CMO98].

5 Scalar Multiplication with Direct Computation of $2^k P$

By using our previous formulae for direct computation of $2^k P$, we can improve elliptic scalar multiplication with the sliding signed binary window method [Go98, KT92]. For example we apply our new formulae to the window method with windows of length 4. We represent a scalar m in $P \mapsto mP$ with a *nonadjacent form* (NAF) ¹. For example, $m = (1101110111)_2$ will be represented as $m' = (100\tilde{1}000\tilde{1}00\tilde{1})_{NAF}$, where $\tilde{1}$ denotes -1 .

¹ Koyama and Tsuruoka pointed out that NAF is not necessarily the optimal representation to use [Go98, KT92]. Although it has minimal weight, allowing a few adjacent nonzeros may increase the length of zero runs, which, in turn, reduces the total number of multiplies. Their method may be useful for our scalar multiplication with direct computations of $2^k P$.

Algorithm 4 describes scalar multiplication on elliptic curves using our direct computations of $2^k P$ in the case of k up to 4.

Algorithm 4: Elliptic scalar multiplication using our direct computation of $2^k P$ in the case of k up to 4

INPUT: $P \in E(\mathbb{F}_p)$, $m \in \mathbb{Z}$

OUTPUT: $mP \in E(\mathbb{F}_p)$

Step 1. Construct NAF representation

$$m = (e_t e_{t-1} \cdots e_1 e_0)_{NAF}, \\ e_i \in \{-1, 0, 1\}$$

Step 2. Precomputation

$$2.1 \quad P_6 \leftarrow 6P$$

2.2 For i from 7 to 10 do:

$$P_i \leftarrow P_{i-1} + P$$

Step 3. $P_m \leftarrow \mathcal{O}$, $i \leftarrow t$

Step 4. While $i \geq 0$ do the following:

4.1 If $e_i = 0$ then:

find the longest bitstring $e_i e_{i-1} \cdots e_l$ such that $e_i = e_{i-1} = \cdots e_l = 0$,
and do the following

$$P_m \leftarrow 2^{i-l+1} P_m$$

$$i \leftarrow l - 1$$

4.2 else ($e_i \neq 0$):

If $(e_i e_{i-1} e_{i-2} e_{i-3})_{NAF} > 0$ then:

$$P_m \leftarrow 16P_m + P_{(e_i e_{i-1} e_{i-2} e_{i-3})_{NAF}}$$

else:

$$P_m \leftarrow 16P_m - P_{|(e_i e_{i-1} e_{i-2} e_{i-3})_{NAF}|}$$

$$i \leftarrow i - 4$$

Step 5. Return P_m

In Algorithm 4, we compute $16P$ directly from P in each window instead of 4 separate doublings. In **Step 4.1** with strings of zero-runs in the scalar m_{NAF} , we should choose computations $16P$, $8P$, $4P$ or $2P$ optimally. This can be done with rules such as: 1) If a length of zero equals to 4, we compute $16P$. 2) If a length of zero equals to 3, we compute $8P$, and so on.

Using our algorithms for scalar multiplication, many of the doublings in an ordinary window method will be replaced by the direct computation of $16P$. Therefore, if one computation of $16P$ is relatively faster than four doublings, scalar multiplication with our method may be significantly improved. We will examine this improvement by real implementation in the next section.

6 Implementation

In this section, we implement our previous methods. We implement 6 elliptic curves over $\mathbb{F}_p$ with $\log_2 p = 160, 192, 224, 256, 384$ or 521 . We call these curves

P160, P192, P224, P256, P384 or P521, respectively. Their parameters can be found in A, and they all have prime order. Moreover, the coefficients a of the curves are chosen as equal to 1 or -3 for efficient implementation. However, in our implementation, for the purpose of making comparison with general curves, we apply general algorithms that do not depend on a parameter a .

The platform was a Pentium II 400MHz, Windows NT 4.0 and Watcom 11.0. Programs were written in assembly language for multi-precision integer operations, which may be time critical in implementation, or in ANSI C language for other operations.

6.1 Inversion

We applied Lehmer's method [LE38] to modular inversion. In affine coordinate computations, modular inversion in $\mathbb{F}_p$ is more expensive than modular multiplication. Therefore, it is important to apply an efficient method to modular inversion.

Remark 1. We applied Lehmer's method to field inversion in our implementation. Although several methods of inversion have been developed, it is not clear which method delivers the most practical speed increase for the range of integers that interests us (up to 521-bit). If we have an efficient method for inversion, elliptic curve cryptosystems with affine coordinates may be faster than those with projective or weighted projective coordinates.

In our implementation, the speed ratio of a multiplication to an inversion I/M is approximately equal to 25 in cases of $\log_2 p = 160$. Our I/M tends to decrease with larger $\log_2 p$.

6.2 Number of $2^k P$ Computations in the Window Method

Table 2 shows the number of computations of $2^k P$ and additions required in the sliding signed binary window method based on Algorithm 4. Typical window sizes of 2 and 4 were used. The numbers were counted by our implementation. In the case of a window of length 2, direct computations of $4P$ can be used. In the case of a window of length 4, direct computations of $4P$, $8P$ and $16P$ can be used.

We can see from the table that: 1) With direct computations of up to $16P$, the computational efficiency of $16P$ significantly affects scalar multiplication. 2) With direct computations of up to $4P$, the computational efficiency of $4P$ significantly affects scalar multiplication.

6.3 Timings

Table 3 shows timings of point additions and direct computations of $2^k P$ in our implementation. Table 4 shows timings of scalar multiplication. In Table 4, "traditional" means a sliding signed binary window method using addition and

Table 2. Number of computations of $2^k P$, where $k = 1, 2, 3$ or 4 , and additions in the sliding signed binary window method with window length 2 or 4

Curves	Add.	Window of length 4				Window of length 2	
		$2P$	$4P$	$8P$	$16P$	$2P$	$4P$
P160	36.82	14.93	4.99	2.58	31.75	17.51	71.07
P192	37.63	15.29	5.06	2.64	39.54	17.93	86.78
P224	37.77	15.31	5.05	2.69	47.49	18.00	102.72
P256	41.77	15.31	5.06	2.70	55.53	18.01	118.82
P384	48.76	17.13	5.10	3.59	86.26	22.72	181.21
P521	90.39	31.34	14.51	6.94	109.87	39.28	241.19

doubling. These two tables also show timings for modified jacobian coordinates. In the case of affine coordinates, we used direct computation of $4P$, $8P$ and $16P$. In the case of weighted projective coordinates, we used direct computation of $4P$.

As can be seen from Table 4, in the case of affine coordinates, in which direct computation of $2^k P$ with $2 \leq k \leq 4$ was used, we achieved a 45 percent performance increase in the scalar multiplication of the elliptic curve of size 160-bit. In the case of weighted projective coordinates, in which direct computation of $2^k P$ with $k = 2$ was used, we achieved a 15 percent performance increase in the scalar multiplication of the elliptic curve of size 160-bit.

In the case of modified jacobian coordinates, we have not yet developed formulae for direct computation of several doublings. Our experimental results suggest that direct computation provides a performance increase. Therefore, once efficient formulae are available, modified jacobian coordinates may provide superior results.

We have one other observation to make from Table 4. In most implementations of curves of size 160-bit, scalar duplications in projective or weighted projective coordinates are faster than those in affine coordinates. On the other hand, in our implementation with the new method, when $\log_2 p$ is relatively large (P384 and P521), scalar multiplications in affine coordinates are faster than those in weighted projective coordinates. As previously stated, the answer to the question “which coordinate system is faster” strongly depends on the ratio I/M . In our implementation, I/M decreases with larger $\log_2 p$. This is the main reason that scalar multiplication in affine coordinates is faster in cases of P383 and P521.

7 Conclusions and Further Work

In this paper, we have constructed formulae for computing $2^k P$ directly from $P \in E(\mathbb{F}_p)$, without computing the intermediate points $2P, 2^2P, \dots, 2^{k-1}P$ in terms of affine and weighted projective coordinates. We showed that our algorithms are more efficient than k separate doublings and lead to a running time improvement of scalar multiplication. Combining our method with the mixed

Table 3. Timings of a point addition and direct doubling in msec. (Pentium II 400MHz)

Curves	Affine					Weighted Projective			Modified Jacobian	
	Add.	2P	4P	8P	16P	Add.	2P	4P	Add.	2P
P160	0.140	0.141	0.194	0.243	0.291	0.0582	0.0373	0.0729	0.0714	0.0355
P192	0.151	0.156	0.229	0.284	0.337	0.0827	0.0538	0.105	0.103	0.0502
P224	0.158	0.167	0.256	0.325	0.390	0.0963	0.0630	0.127	0.125	0.0563
P256	0.172	0.175	0.279	0.345	0.415	0.110	0.0713	0.139	0.142	0.0623
P384	0.219	0.229	0.392	0.497	0.616	0.198	0.123	0.238	0.244	0.108
P521	0.358	0.378	0.695	0.892	1.179	0.347	0.214	0.407	0.423	0.180

Table 4. Timings of a scalar multiplication of a random point in msec. (Pentium II 400MHz)

Curves	Affine		Weighted Projective		Modified Jacobian
	Traditional	Proposed	Traditional	Proposed	Traditional
P160	26.8	18.4	9.1	7.9	8.2
P192	35.2	22.5	15.5	13.1	13.2
P224	42.1	27.8	20.6	18.2	18.3
P256	50.9	34.5	25.7	22.3	22.8
P384	98.3	65.7	68.2	58.5	59.0
P521	215	148	163	139	140

coordinate strategy proposed in [CMO98] may increase elliptic scalar multiplication performance.

References

CC86. D. V. CHUDNOVSKY and G. V. CHUDNOVSKY, “Sequences of numbers generated by addition in formal groups and new primality and factorization tests”, *Advances in Applied Math.*, **7** (1986), 385–434.

CMO98. H. COHEN, A. MIYAJI and T. ONO, “Efficient Elliptic Curve Exponentiation Using Mixed Coordinates”, *Advances in Cryptology – ASIACRYPT’98*, LNCS, **1514** (1998), Springer-Verlag, 51–65.

Go98. D. M. GORDON, “A survey of fast exponentiation methods”, *Journal of Algorithms*, **27** (1998), 129–146.

GP97. J. GUAJARDO and C. PAAR, “Efficient Algorithms for Elliptic Curve Cryptosystems”, *Advances in Cryptology – CRYPTO’97*, LNCS, **1294** (1997), Springer-Verlag, 342–356.

HT99. Y. HAN and P.C. TAN, “Direct Computation for Elliptic Curve Cryptosystems”, Pre-proceedings of CHES’99, (1999), Springer-Verlag, 328–340.

IEEE. IEEE P1363/D11 (Draft Version 11), (1999), <http://grouper.ieee.org/groups/1363/>

ITTTK98. K. ITOH, M. TAKENAKA, N. TORII, S. TEMMA and Y. KURIHARA, “Fast Implementation of Public-key Cryptography on a DSP TMS320C6201”, *Cryptography Hardware and Embedded Systems*, LNCS, **1717** (1999), Springer-Verlag, 61–72.

- Ko87. N. KOBLITZ, “Elliptic curve cryptosystems”, *Mathematics of Computation*, **48** (1987), 203–209.
- KT92. K. KOYAMA and Y. TSURUOKA, “Speeding up Elliptic Cryptosystems by Using a Signed Binary Window Method”, *Advances in Cryptology – CRYPTO’92*, LNCS, **740** (1993), Springer-Verlag, 345–357.
- LE38. D. H. LEHMER, “Euclid’s Algorithm for Large Numbers”, *American Mathematical Monthly*, **45** (1938), 227–233.
- Mi85. V. MILLER, “Uses of elliptic curves in cryptography”, *Advances in Cryptology – CRYPTO’85*, LNCS, **218** (1986), Springer-Verlag, 417–426.
- MOC97a. A. MIYAJI, T. ONO and H. COHEN, “Efficient Elliptic Curve Exponentiation (I)”, *Technical Report of IEICE*, ISEC97-16, (1997)
- MOC97b. A. MIYAJI, T. ONO and H. COHEN, “Efficient Elliptic Curve Exponentiation”, *Advances in Cryptology – ICICS’97*, LNCS, **1334** (1997), Springer-Verlag, 282–290.
- MU97. V. MÜLLER, “Efficient Algorithms for Multiplication on Elliptic Curves”, *Proceedings of GI-Arbeitskonferenz Chipkarten 1998*, TU München, (1998)
- NIST. NIST, “Recommended Elliptic Curves for Federal Government Use”, <http://csrc.nist.gov/encryption/>, (1999)
- So97. J.A. SOLINAS, “An Improved Algorithm for Arithmetic on a Family of Elliptic Curves”, *Advances in Cryptology – CRYPTO’97*, LNCS, **1294** (1997), Springer-Verlag, 357–371.
- WMPW98. E. DE WIN, S. MISTER, B. PRENEEL and M. WIENER, “On the Performance of Signature Schemes Based on Elliptic Curves”, *Algorithmic Number Theory III*, LNCS, **1423** (1998), Springer-Verlag, 252–266.

A Curves

In this appendix, the curves that we have implemented are given. P192, P224, P256, P384 and P512 are given in [NIST] as recommended curves.

P160

$p = 736459809436275298942988210292996840747673059329$

$a = 1$

$b = 148564875525137510302727914159578416202993714965$

$\#E(\mathbb{F}_p) = 736459809436275298942987873098523465705697104451$

High Performance Agile Crypto Modules

Chandana G. Gamage¹, Jussipekka Leiwo², and Yuliang Zheng¹

¹ Monash University, PSCIT, McMahons Road, Frankston, Vic 3199, Australia,
{chandag,yuliang}@infotech.monash.edu.au

² Vrije Universiteit, Division of Sciences, De Boelelaan 1081A, 1081 HV Amsterdam,
The Netherlands, leiwo@cs.vu.nl

Abstract. This paper examines the impact of the primary symmetric key cryptographic operation on network data streams, encryption of user data, have on the overall traffic throughput. The encryption function which provides the basic confidentiality security service is studied from two cryptographic design perspectives based on the ability of a network cipher unit to dynamically re-parameterize itself while processing a high speed data stream. The designs studied in this paper were chosen based on their suitability for high speed network operation and their flexibility in satisfying dynamic security requirements. We develop analytical techniques to model the performance of each scheme.

Keywords. Network security, High performance, ATM networks, Performance modelling

1 Introduction

In secure communication networks with an end-to-end security association, the basic security service of data confidentiality is provided by a symmetric key cipher. The two basic classes of symmetric key ciphers are the stream cipher and block cipher. Stream ciphers encrypt individual characters in a unit of plaintext one character at a time. For network communication, this generally means bit-by-bit encryption. In contrast, block ciphers such as DES [11] encrypt a fixed size group of characters at a time. Stream ciphers are useful in a communication network that does not buffer protocol transfer units but process a continuous bit stream. However, this is not the case with modern network designs which utilize a fixed length protocol transfer unit commonly referred to as a packet or a cell and variable length transfer units termed segments consisting of a number of fixed size packets or cells. Therefore, in network security, our discussion is limited to encryption function provided by symmetric key block ciphers. Apart from its suitability for use in providing data confidentiality packet oriented data networks, the cryptographic function of block encryption is a central element in providing several other network security services such as data integrity and message authentication. A single block encryption algorithm can be used in the construction of both the encryption function and the message authentication code (MAC) function [12] to provide data confidentiality and data authentication respectively as shown in figure 1. A MAC value simultaneously provides both data integrity and data origin authentication.

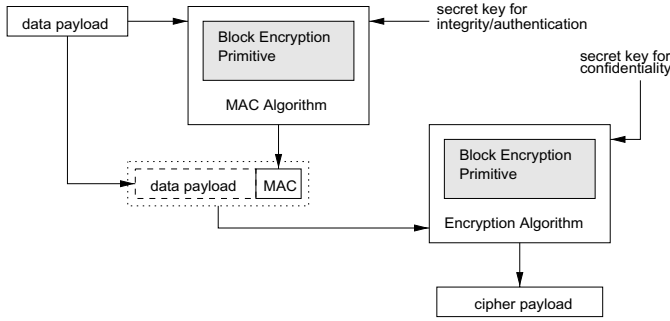


Fig. 1. A function block schematic for generating an encrypted payload with a MAC authenticator value

In the next section, we discuss specific network quality of service (QOS) parameters of interest for analyzing performance of secure communication channels. Thereafter, section 3 presents key-agile encryption technique and section 4 presents algorithm-agile encryption technique. The main reasons for selecting these two encryption techniques are given in section 3.1 and section 4.1, respectively. Apart from the efficiency and flexibility of operation provided by these two schemes in providing basic security services at the network layer of high speed communication networks, they have been used in several research projects [20, 22] to build secure ATM test networks. This is significant for secure high speed network design as ATM is the preferred network technology for multi-service broadband networks (B-ISDN) [13]. The analytical model and numerical example for key-agility (in sections 3.4 and 3.5) and algorithmic-agility (in sections 4.2 and 4.3) provide a basis and justification for use of these techniques in high speed network implementation. The paper concludes with remarks on the effect of several other QOS parameters on secure network performance.

2 Secure Communication and QOS Parameters

In the design of high performance network security systems, a detailed analysis of capacity requirements and deliverable throughput is essential. For end-to-end network performance modelling analysis, capacity is represented by the product of number of simultaneously active *secure channels* and the average channel bandwidth which determine the quantity of traffic that the network can sustain. Similarly, throughput is represented by bounds on the number of allowable active channels and allowable transmission losses which determines the amount of traffic successfully transmitted [10]. Inadequate or incomplete analysis of available and required system capabilities could easily lead to vastly under-performing systems. For example, over-estimation of required processing capacity of a security module could result in a conservative design approach that implements only modest and potentially inadequate security capabilities. Similarly, under-estimation

of required processing capacity could result in secure systems that fail to provide expected performance guarantees. This is a particularly important issue in multi-service high speed networks that have been designed to negotiate and then provide a guaranteed QOS for network users. In this respect, we use ATM networks as the basis for discussing the relationship between network QOS and cryptographic performance in secure high speed networks. The ideas presented and results derived in this paper are applicable to other types of networks with QOS support such as TCP/IP networks operating under resource reservation protocol (RSVP) [24]. As the main focus of the study presented in this paper is the impact of symmetric key cryptographic techniques on secure real-time data transmission, the QOS parameters we consider relate to data transfer and not to call control for the connection-oriented ATM networks. The QOS parameters of concern are bit error ratio (BER), cell loss ratio (CLR) and cell insertion ratio (CIR).

BER is defined as the ratio between number of bit errors that occur in a transmission system and the total number of bits transmitted and is mainly dependent on the transmission system being used including its physical characteristics (such as for copper, fiber optic, etc.) and the operational environment (such as electro-magnetic interference). The use of optical fiber technology in high speed networking has greatly reduced the expected value of BER. If the bit error occurrence in a link is a random process (as the case in modern high speed optical links), then the probability of an error free transmission of an ATM cell is $(1 - \text{BER})^{384}$. Here, only the cell payload of 48 bytes (384 bits) is considered for bit error detection as an unrecoverable bit error in the cell header portion will result in the cell not being allocated to any particular stream.

CLR is defined as the ratio between number of cells lost in transmission and the total cells transmitted over a period of time. The main reasons for specifying a CLR for ATM network connections are the cell discard due to buffer overflow and unrecoverable bit errors in cell headers. For our analysis, both BER and CLR can be considered as a single factor affecting performance as cells with bit errors in encrypted payloads also need to be discarded on integrity check failure at the same layer on which decryption is done.

The cell insertion occurs due to bit errors in the header causing mis-routing of cells onto wrong channels when the error in header address field matches with a correct switching label at a ATM node. CIR is defined as the ratio between number of cells misrouted to a destination and the total number of cells delivered to that destination address. CIR also causes loss of cryptographic synchronization and discarding of several cells. Thus, in the rest of the paper, our reference to CLR actually refers to compound effect of cell losses due to BER, CLR and CIR under encrypted cell transmission.

For secure cell transmission, we may consider a cell with even a single bit error as a lost cell as a single bit error in an encrypted payload can expand randomly on decryption of the cell causing bit error expansion within a cell. This intra-cell error expansion will make any use of error correction codes largely ineffective. Depending on the mode of encryption [12] used and the construction of the

encryption unit, bit errors may propagate to adjacent cells also. For example, if the cipher block chaining (CBC) or cipher feedback (CFB) mode is used, then a bit error within one encrypted cell payload will spread through rest of the cell stream on decryption. While the output feedback (OFB) mode does not cause bit errors in the cipher payload to spread, this mode requires periodic synchronizing of the encryptor and decryptor unit (say, by using a special marker cell) to recover from possible cell losses. Both the CBC and CFB modes are self synchronizing with cell loss propagation limited to only one additional block. The other common mode of operation, electronic codebook (ECB) has only intra-block bit error propagation and the cell stream is self synchronizing on cell losses with no loss effect propagation. However, ECB is not recommended for use in many applications as it is vulnerable to both substitution or reordering attacks and cryptanalytic attacks due to repeating plain text blocks producing identical cipher text blocks. For data networks, any unrecoverable errors in transmission of a protocol unit at a given layer (for example, a packet at network layer) detected at a receiver usually invokes an error recovery procedure at the same layer or at a layer above. As the standard error recovery mechanisms is to request retransmission of the entire protocol unit of transfer, the benefits of limited error propagation or self synchronization of an encryption algorithm is of limited value in network applications.

In summary, the major effect of a cell that was lost or was in error is the loss of cryptographic synchronization for adjacent cells in a stream of encrypted cells for widely used encryption modes resulting in more than one cell being lost and potentially a larger block of cells mapping to an upper level protocol data unit (PDU) being discarded.

3 Key-Agile Encryption

3.1 Encryption at Physical Layer

One of the simplest method to secure communication between two network end-points is to agree on a cryptographic context including a symmetric key for traffic encryption using off-line mechanisms. This allows host-level security management at end-points. Thereafter, confidentiality and integrity services for the user data part of a traffic stream can be delivered through encryption and data integrity functions at the highest possible speed at the physical or link layer as no further on-line security related negotiations take place. The main disadvantage of this approach is that it is not possible to secure the network traffic at a finer granularity such as per-user or per-application. Also, the static pre-configuration of cryptographic keys and consequent long-term key usage allows an attacker to capture a large amount of ciphertext messages encrypted with the same key (especially in a high speed network) which could facilitate an off-line cryptanalytic attack. Furthermore, as inter-networking units such as switches and routers need to process header information contained in a protocol transfer unit, the full traffic stream cannot be encrypted. If the header information also need to be protected, for example, to prevent traffic analysis, then the traffic streams need

to be secured on hop-by-hop basis with decryption at input and re-encryption at output in each inter-networking unit using the symmetric key shared between each unit. In addition to the disadvantages mentioned earlier, this makes the switching and routing nodes in a network highly vulnerable to attacks. Also, the amount of cryptographic processing required at each inter-networking unit would be excessively high resulting in lower network throughput performance. The type of network security provided by such a scheme would be public network-to-network rather than private user-to-user as the management of cryptographic keys in the network infrastructure would be outside end-user control. Therefore, even with the potential for fast implementation through hardware and transparency of operation, this type of physical layer or link layer schemes for securing traffic is not suitable for modern high speed networks.

With respect to ATM networks, the physical layer or link layer type secure communication would be the use of a single cryptographic context and an associated symmetric key to encrypt the cell stream through a physical link connecting two adjacent ATM switches or through all the VCs between two ATM connection end-points. As this solution essentially creates a secure virtual tunnel that operates at the physical layer allowing fast encryption of cell streams between the end-points, designers of secure networks does not have to consider the behaviour of actual traffic sources (such as continuous or bursty traffic) or protocol semantics of the cell stream (such as native ATM traffic that have cell continuity or IP over ATM type traffic that contain cell blocks). Again, as the scheme requires static keying of physical links, this approach is not suitable for use by ATM end-points that connect through public ATM infrastructure for wide area connectivity. Also, even in the case of a private ATM LAN, this scheme does not have the capability to distinguish between traffic streams requiring security services and those that have no security requirements. Therefore, the potential for computational and transmission resource wastage is quite high. Due to these drawbacks, it is necessary to consider a more flexible and secure scheme to protect ATM cell stream at the next level of granularity, that is, for the individual VC.

3.2 Encryption at a Virtual Layer

Protection of data communication at a virtual layer as opposed to the physical layer, involves identification of end-to-end flow of protocol transfer units and applying cryptographic operations to these flows separately using individually specified security contexts. In network protocol models that provide an end-to-end transport connection layer, this mode of operation can be implemented with flow identification by the connection label embedded in data packets (for example, TCP source and destination port address gives a flow identifier at transport layer). If the network protocol model only provides a datagram style hop-by-hop virtual layer, the packet flow differentiation has to be done using a combination of header information (for example, the IPv6 source and destination addresses, identification, protocol and options fields together defines a unique

flow label. In IPv6, the flow label header field combined with the two IP addresses provide the flow identifier).

For ATM networks, the term key-agile cell encryption refers to the scheme by which the traffic through each individual VC is secured using a unique short-term symmetric key [20]. Among the many advantages of this approach are the ability to dynamically negotiate a shared key for each VC at its connection setup time, ability to determine symmetric key characteristics such as the key length based on end-point security requirements and frequency of dynamic key updates for long-lived VCs. However, to achieve key agility, several changes are required for the basic cell relay function at the ATM layer. The two main changes required are:

1. Table look-up of the associated symmetric key at each end-point on a per-cell basis.
2. Execution of cryptographic operations including integrity checksum calculation and encryption (resp. checksum verification and decryption) on the cell payload.

As the cryptographic operations are performed only on the cell payload and header is sent in clear text, this cryptographic table look-up is required only at end-point ATM nodes. The standard ATM layer functions such as cell relay, header recalculation and queue management at intermediate switching nodes are not affected by this type of secure cell transmission.

3.3 Dynamic Key Look-Up

Each active VC in an ATM network is uniquely identified by a 24-bit combined virtual path identifier (VPI) and virtual channel identifier (VCI) address at the user network interface (UNI). As the particular symmetric key to be used for securing a cell payload is determined by this VPI/VCI pair, cryptographic operations can be done only after a payload had its header prefixed. Also, the ATM layer performs cell-level address-based statistical multiplexing using the VCI in each cell. Therefore, in theory, key-agile cell encryption or decryption may require a table look-up for each cell received at an incoming or outgoing port. As the symmetric keys are much longer than ATM address fields (minimum of 64-bit and up to 256-bit long keys), the usual technique is to use a dual table solution in which a larger table indexed by VPI/VCI values contain pointers to a smaller cryptographic key table [20].

This symmetric key retrieval from a potentially very large table can therefore become a performance bottleneck in a high speed key-agile cell encryption scheme. There are several techniques to limit the negative impact on the performance:

1. *Caching of recently or frequently used symmetric keys.* This standard performance enhancement technique can be used by directing the dual table key access operation through a small cache of (VPI/VCI, symmetric key) tuples stored in high speed memory.

2. *Restricted range of VCIs for secure connections.* In the ATM standard, there are several blocks of VPI/VCI combinations that are reserved for specialized use. Among these are the address labels used for unassigned cells processed by the transport layer, cells introduced by the physical layer, idle cells for cell boundary resynchronization and OAM cells. Similarly, a range of VPI/VCI values could be defined for use in secure VC establishment by end-points. This approach will essentially limit the size of symmetric key table for fast access.
3. *Look-up table implementation in content addressable memory (CAM).* A look-up table constructed using CAM allows parallel search of its index entries, thus speeding up the content retrieval.

Although above techniques to reduce the performance penalty in cryptographic table look-up appear to be costly to implement (for cache and CAM) and arbitrarily restrictive (for limited VCI range), in actual high speed networks the table size for active VCs may be much smaller than the theoretical maximum possible due to other factors such as limited buffer space at multiplexors and allowed cell loss ratio (CLR). Therefore, before designing a costly or restrictive symmetric key table look-up scheme, it is important analyze the network operation to determine the actual performance requirements. Results of such an analysis could direct secure system designers to more affordable and realistic solutions.

3.4 An Analytical Model of the Key Agile Network Port

Consider N VCs multiplexed at an outgoing port with V_n , where $n = 1, \dots, N$, as the transmission rate at time t for a VC labelled with number n . We assume the rates of the N circuits to be independent and identically distributed random variables. Now, we would like to determine the rate r such that the multiplexed link can carry the aggregate traffic of N VCs bounded by the allowable CLR. Alternatively, we could determine the number of traffic sources N for a given average transmission rate of r .

$$P\{V_1 + \dots + V_N > rN\} \leq \text{CLR} \quad (1)$$

To model fairly generic network traffic conditions, we further assume the input sources of the VCs to be homogeneous on-off sources with $P(\text{on}) = p$, peak rate a and the number of VC input sources in *on* state to have a binomial distribution. Then the probability that the aggregate transmission rate of active VCs exceeds the output rate of the outgoing link can be represented as

$$P\{V_1 + \dots + V_N > rN\} = \sum_{n \geq \frac{Nr}{a}}^N \binom{N}{n} p^n (1-p)^{N-n} \quad (2)$$

Using the Bahadur-Rao theorem [3,23] we can determine an approximation for the above equation 2 as follows that would allow us to compute a value for N .

$$P\{V_1 + \dots + V_N > rN\} \approx \frac{1}{\sqrt{2\pi}\sigma\theta_r\sqrt{N}} \times e^{-NI(r)} \quad (3)$$

where

$$\begin{aligned} \theta_r &= \frac{1}{a} \log \left(\frac{r(1-p)}{p(a-r)} \right), \\ I(r) &= \frac{r}{a} \log \left(\frac{r(1-p)}{p(a-r)} \right) - \log \left(\frac{a(1-p)}{(a-r)} \right) \text{ and} \\ \sigma^2 &= r(a-r) \end{aligned}$$

3.5 A Numerical Example

To illustrate the operation of a high speed network under the above modelling assumptions, consider a port with VCs having a peak rate $a = 10$ Mbps (say, switched LAN interfaces) and rate $r = 5.33$ Mbps. Under these conditions, for $\lambda = 1/3$ and $\mu = 1/2$ gives $P(on) = p = 0.40$ at a mean rate $p \times a = 4$ Mbps. Now, the parametric values of equation 3 can be computed as $\theta_r = 0.05377$, $I(r) = 0.03597$ and $\sigma^2 = 24.89$. If the CLR is assumed to be 10^{-9} (for fiber optic links), the probability $P\{V_1 + \dots + V_N > rN\}$ holds for $N \geq 500$.

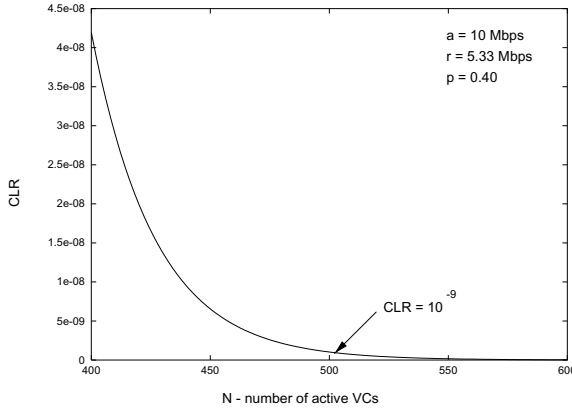


Fig. 2. A numerical example showing the limiting effect of CLR on the number of active VCs multiplexed onto a single ATM link

The results of the above numerical example show, that for an ATM outgoing port with an aggregate capacity of $N \times r = 2665$ Mbps (approximately an OC-48 link), the requirement is to maintain a cryptographic table with a maximum of 500 entries only as the number of active VCs are bounded by the CLR (see graph in figure 2). Also, more importantly, the port will have to look-up keys only 500 times every second or once every 2 millisecond given the common input source behaviour as on-off models. It is interesting to compare these performance values computed above with a simple derivation of key switching for an OC-48 link.

In this case, if we assume cell-by-cell key switching, a table look-up is required every $(424 \text{ bits} / 2488 \text{ Mbps}) = 170 \text{ nanoseconds}$.

Above example clearly shows the unnecessarily high performance bounds a designer would obtain from a strictly theoretical derivation (a key look-up every 170 ns) against a more realistic value obtained by stochastic modelling (a key look-up every 2 ms). In real-time operation, the performance of the crypto unit is not limited by the average processing but by the peak instantaneous loads. As we have incorporated this behaviour into the analytical model through the peak rate a , we can conclude that key-agile cell encryption is a practical technique for implementing flexible security schemes for high speed networks.

4 Algorithm-Agile Encryption

4.1 Need for Algorithmic Context Negotiation

A high speed network that has only limited operational requirements and policies with regard to securing its data transmissions can standardize on a pre-agreed set of cryptographic parameters including algorithms for encryption and modes of operations, algorithms for digital signatures, key lengths, key update and access control policies and other cryptographic variables. Thereafter, the network can provide a secure per-connection end-to-end data transmission mechanism such as the key-agile cell encryption previously described in section 3 when only the session key is dynamically selected or updated. However, for a multi-service network spanning many operating and regulatory environments, a cell encryption mechanism based on a single algorithmic context (or few algorithmic contexts that are pre-configured for specific static connections) is clearly inadequate. When end-systems located in different operational and regulatory environments setup a secure connection, it is necessary to dynamically negotiate a cryptographic context including different algorithms acceptable to both end-users. For ATM networks, the industry standard security specification document [2] proposed by the ATM forum includes security signalling at connection setup time to carry out this task and provide adaptive secure data transmission. This need for dynamic cryptographic context selection is applicable to other network protocols that provide an end-to-end connection at a virtual layer such as the transport layer of TCP/IP protocol stack.

As a concrete example of the need for algorithmic agility, consider the export control regulation for cryptographic products enforced by many countries. While many countries allow strong cryptographic security (such as symmetric keys of 128 bit length or longer) within the national boundaries, strength of external secure communication is arbitrarily limited (such as short 40 bit keys). Use of different key lengths for the same algorithm itself constitutes algorithmic agility since the crypto units need to be reconfigured at real-time with new values associated with the key schedule. The situation becomes even more complex when multiple keys (such as in triple DES) are used to circumvent key length restrictions. Therefore, cell-level encryption devices in high speed trans-national

networks need to be designed with an array of crypto units that can operate in parallel to process multiple data streams simultaneously.

The term algorithm-agile cell encryption refers to the scheme by which traffic through different VCs may be processed according to a dynamically negotiated cryptographic context. The most commonly cited approach in literature is to implement a cryptographic module with multiple parallel processing units that are dynamically loaded with associated algorithmic context for the cell being processed currently in each unit [22,21].

However, in this paper, we consider using crypto units that are pre-configured for different algorithms and associated crypto variables. This method allows easier combination of key agility and algorithmic agility within a single security module. In the security module, one multiplexor unit can differentiate cells according to their algorithmic-context and feed to appropriate crypto unit while within that unit per-cell key and other related crypto variable look-up can be done.

The objective of the modelling and analysis done in the remainder of this section is to show the practicality of algorithm-agile cell processing using above described type of crypto units for normal high speed network operations under typical traffic loads and for reasonable number of algorithmic-context options. For this purpose, first we develop a system model based on the operational properties of an algorithm-agile crypto module. While the concept of algorithmic agility is applicable to any type of protocol scheme allowing end-to-end network connections in which users want dynamically specify a security context, the analytical model is developed for ATM, which is the most common type of high speed network implementation technology. Thereafter, a numerical example is used to evaluate the practical utility of the scheme under high speed network operational parameters.

4.2 An Analytical Model of the Algorithmic Agile Network Port

Consider an algorithm-agile crypto module as shown in figure 3, that has m number of crypto processor units, each capable of processing cells that have been cryptographically secured according to a specific algorithmic context. Lets assume that each processor unit P_i is capable of processing p_i cells per time unit and that the combined crypto module receives n cells per time unit over its multiplexed input. We further assume the distribution of algorithmic context among arriving cells to be a Poisson distribution owing to its memoryless property. Therefore, if the algorithmic contexts are numbered as $0, \dots, m-1$ then the probability function

$$p(x) = \begin{cases} \frac{\lambda^x e^{-\lambda}}{x!} & \text{if } x \in \{0, \dots, m-1\} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

gives the distribution of algorithmic contexts among the cells that have arrived during a sample time unit. Here, λ is the mean value that determines the shape of the algorithmic context distribution.

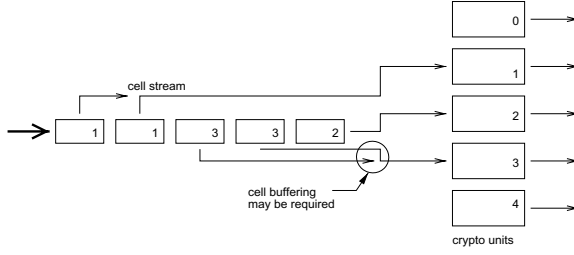


Fig. 3. Model of algorithmic agile cell processing

If we assume that each crypto unit P_i takes τ_i time units to process a cell (that is, $p_i \times \tau_i = 1$), then the amount of time T_i spent by each crypto unit in processing cell payloads is

$$T_i = n p(i) \tau_i \quad \text{for } i = 0, \dots, m-1 \quad (5)$$

As the above model of cell processing consider a time window of single unit, for optimum performance by the crypto processing units without causing cell discard or excessive buffering, following inequality should hold

$$T_i = n \frac{\lambda^i e^{-\lambda}}{i!} \tau_i \leq 1 \quad \text{for } i = 0, \dots, m-1 \quad (6)$$

4.3 A Numerical Example

To illustrate the operation of an algorithmic agile crypto module based on the above model, let's consider a 16 processor system with an OC-48 link capacity that input approximately 5.87 million cells per second (2488Mbps / 424bits) with an average cell processing time of 160ns. The graphs in figure 4 show the processor utilization profiles corresponding to different cryptographically secured cell traffic profiles as determined by various λ values. The numerical values chosen for the analysis conform to the optimality bounds given by equation 6.

The average value of 160ns to process the secured payload of a cell requires a sustained throughput of 2400Mbps (384bits / 160ns), which may seem difficult to achieve given current state of the art in VLSI based cryptographic processor cores. However, the throughput requirement can be easily reduced without an adverse effect on the overall system performance. Consider the crypto unit utilization graph for $\lambda = 2$ in figure 4 which represent a typical crypto traffic profile scenario with only 2 or 3 heavily used algorithmic contexts and the remaining modules used less frequently. For this particular case, the average value of τ can be increased from 160ns to 640ns while still remaining within the bounds set in equation 6. This will result in a required throughput of 600Mbps which is within the range of hardware implementations of widely used symmetric key cryptosystems such as DES [8].

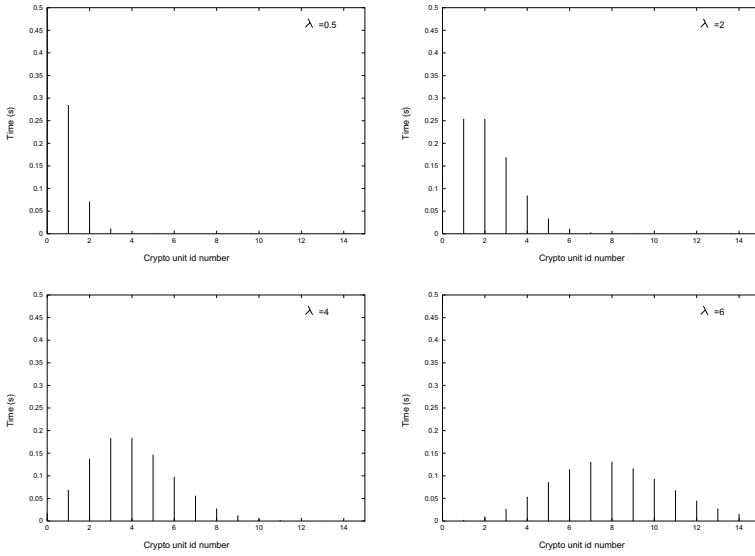


Fig. 4. The utilization of individual units in an algorithm-agile crypto module for different cell traffic profiles

The sample values used in the numerical example shown above are based on commercial VLSI crypto core units such as [1, at 640 Mbps] and [5, at 528 Mbps] that support the DES algorithm [11]. The table 1 gives a summary of the projected throughput requirement in terms of bit encryptions per unit time for different algorithm-context loadings. As newer encryption algorithms such as SPEED [25], Twofish [16] and Rijndael [9] that are much faster than the DES algorithm comes into wider use, achieving the throughput requirements for high speed cell encryption with these new generation algorithms including the next generation Advanced Encryption Standard (AES [4,17]) is likely to be much more practical.

Table 1. Average throughput requirement for crypto units under different traffic profiles ($m = 15$ and $n = 2488Mbps$ are the parameters in equation 6)

λ	τ (ns)	Active units	Throughput (Mbps)
0.5	530	1	725
2	640	2/3	600
4	800	4/5	480
6	1060	6	365

In our model, we have not explicitly considered the possibility of more than one crypto unit supporting the same algorithmic context. In this instance, a

separate scheduler will have to be used to admit cells to individual units in the crypto processor group with the common algorithmic context. This will have no meaningful impact on our analysis which attempt to determine if an overall crypto module consisting of several separate units can process a multiplexed cell stream within performance bounds set by a high speed network environment.

5 Summary

The idea of key-agile encryption and details of a proof-of-concept implementation work done by Stevenson et al. [20] have appeared in literature preceding the work described in this paper. Also, design details of a cryptographic device called a *CryptoNode* incorporating similar ideas have been presented by Chuang in [6,7]. Independent of the work described in this paper, the concept of algorithm-agile encryption and related work has been presented by Sholander et al. [18], Tarman et al. [22,21] and Pierson et al. [14]. The use of a single key block in algorithmic agile systems, proposed by Smart [19], to assist in the rapid real-time algorithm selection can be combined with algorithm-agile crypto units to further improve performance.

To obtain operating parameters for the design of high performance security modules (specifically, high speed cell encryptors), designers can utilize any combination of analytical models, simulations and test implementations. Although both simulations and test-beds can benefit from results obtained through analytical models, there is a clear lack of work in this area. As real performance of high speed networks continue to increase and wide area networks grow in complexity, both simulation and test implementation will be more difficult. Therefore, analytical mechanisms such as ours that examines the performance of secure systems will be much valued tool for system designers.

Two other important data transfer QOS parameters are the end-to-end transfer delay (CTD) and the cell delay variation (CDV) or jitter. The analysis in this paper considered crypto modules to be delay units of fixed time duration that increase end-to-end transfer delay by a pre-calculatable value and thus cause no change in original negotiated CDV. However, as different crypto units within the security processing module are most likely to have different latencies due to differences in algorithms and key lengths, it will cause CDV in the multiplexed cell stream. Therefore, above modelling assumptions will be true only if delay equalization is done at the processing module at the cost of increased CTD. Otherwise, original QOS negotiation at connection setup time needs to consider the added CDV due to security related processing. To develop analytical models that can accurately represent the real-time behaviour of high performance secure networks, above security related delay analysis must be complemented with other factors such as connection admission control policies, job scheduling at security modules (processors) and management of shared resources such as buffers.

As the agile crypto units are designed for block-mode operation in uniquely identified end-to-end data flows, they can be positioned at several of the layers

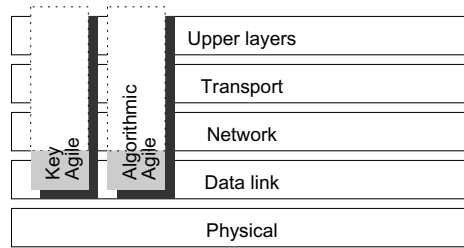


Fig. 5. The positioning of interfaces to the agile crypto units in a network protocol stack

in a network protocol architecture that have a defined protocol unit. However, as shown in figure 5, agile crypto units should be located nearer to the portion of a network protocol stack that is implemented in hardware to achieve the highest possible performance.

References

1. The SafeNet ADSP2141L from Analog Devices, Inc, March 1999. Available from <http://www.analog.com/>.
2. The ATM Forum, Mountain View, CA. *ATM Security Specification, Version 1.0*, af-sec-0100.000 edition, February 1999.
3. R. R. Bahadur and R. R. Rao. On deviations of the simple mean. *Annals of Mathematical Statistics*, 31(2):1015–1027, 1960.
4. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay. Report on the AES candidates. In *Second Advanced Encryption Standard (AES) Candidate Conference*, Rome, Italy, March 1999. NIST.
5. CDI 2100 cryptographic processor from Cognitive Designs, Inc, May 1999. Available from <http://www.cognitive-designs.com/>.
6. S.-C. Chuang. Securing ATM networks. In *Proceedings of the Third ACM Conference on Computer and Communications Security (CCS'96)*, pages 19–30, New Delhi, India, March 1996. ACM Press.
7. S.-C. Chuang. Securing ATM networks. *Journal of Computer Security*, 4:289–329, 1996.
8. L. Claesen, J. Daemen, M. Genoe, and G. Peeters. Subterranean: A 600 Mbit/sec cryptographic VLSI chip. In E. Straub, editor, *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'93)*, pages 610–613, Cambridge, MA, October 1993. IEEE Computer Society Press.
9. J. Daemen and V. Rijmen. The block cipher Rijndael. In *Proceedings of CARDIS'98: Third Smart Card Research and Advanced Applications Conference*, Lecture Notes in Computer Science, Louvain-la-Neuve, Belgium, September 1998. Springer-Verlag.
10. G. N. Higginbottom. *Performance Evaluation of Computer Networks*. Artech House, Inc, Norwood, MA, 1998.

11. National Bureau of Standards, U.S. Department of Commerce. *Data Encryption Standard. Federal Information Processing Standards Publications (FIPS PUB) 46*, January 1977.
12. National Institute of Standards and Technology, U.S. Department of Commerce. *DES Modes of Operation. Federal Information Processing Standards Publication (FIPS PUB) 186*, May 1994.
13. R. O. Onvural. *Asynchronous Transfer Mode Networks: Performance Issues*. Artech House, Inc, Norwood, MA, 1994.
14. L. G. Pierson, E. L. Witzke, M. O. Bean, and G. J. Trombley. Context-agile encryption for high speed communication networks. *SIGCOMM Computer Communication Review*, 29(1):9–9, January 1999.
15. R. A. Rueppel. Stream ciphers. In G. J. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, pages 65–134. IEEE Press, New York, 1992.
16. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, and C. Hall. On the Twofish key schedule. In S. Tavares and H. Meijer, editors, *Fifth Annual International Workshop on Selected Areas in Cryptography (SAC'98)*, volume 1556 of *Lecture Notes in Computer Science*, pages 27–42, Kingston, Ontario, Canada, August 1998. Springer-Verlag.
17. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Performance comparison of the AES submissions. In *Second Advanced Encryption Standard (AES) Candidate Conference*, Rome, Italy, March 1999. NIST.
18. P. E. Sholander and et al. The effect of algorithm-agile encryption on ATM quality of service. In *Proceedings of IEEE GLOBECOM'97*, pages 470–474, Phoenix, AZ, November 1997. IEEE Computer Society Press.
19. N. P. Smart. One key to rule them all. Technical Report HPL-1999-26, Extended Enterprise Laboratory, HP Laboratories Bristol, UK, March 1999.
20. D. Stevenson, N. Hillery, and G. Byrd. Secure communications in ATM networks. *Communications of the ACM*, 38(2):45–52, February 1995.
21. T. D. Tarman, R. L. Hutchinson, L. G. Pierson, P. E. Sholander, and E. L. Witzke. Algorithm-agile encryption in ATM networks. *IEEE Computer*, 31(9):57–64, September 1998.
22. T. D. Tarman, R. L. Hutchinson, P. E. Sholander, R. J. Granfield, L. G. Pierson, P. J. Robertson, and E. L. Witzke. Final report for the robustness-agile asynchronous transfer mode (ATM) encryption laboratory directed research and development project. Technical Report SAND97-2902, Sandia National Laboratories, Albuquerque, NM, December 1997.
23. J. Walrand and P. Varaiya. *High-Performance Communication Networks*. Morgan Kaufmann Publishers, Inc, San Francisco, CA, 1996.
24. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. Rsvp: A new resource reservation protocol. *IEEE Network Magazine*, 7:8–18, September/October 1993.
25. Y. Zheng. The SPEED cipher. In R. Hirschfeld, editor, *Proceedings of Financial Cryptography'97*, volume 1318 of *Lecture Notes in Computer Science*, pages 71–89, Anquilla, BWI, February 1997. Springer-Verlag.

A Three-Party HTTP Proxy to Support Internet Content Regulation

Agung Prasetijo¹, Paul Ashley², Mark Looi¹, Andrew Clark¹, and Gary Gaskell¹

¹ Information Security Research Centre,
School of Data Communications,
Queensland University of Technology,
GPO Box 2434, Brisbane - AUSTRALIA
{agung, mlooi, aclark, gaskell}@isrc.qut.edu.au

² Tivoli, Security Business Unit
9020 Capital of Texas Hwy,
Great Hills Corporate Center,
Building 1, Suite 270, Austin, TX 78759
United States of America
Paul.Ashley@tivoli.com

Abstract. From 1st January 2000, Internet Service Providers in Australia have been required to filter web traffic content. The Australian Broadcasting Authority (ABA) publishes a list of banned web sites and ISPs must enforce this ban. Enforcing Internet content regulation is a security issue: ISPs need to be able to verify the authenticity of a distributed banned list, enforcement will most likely be performed by a security component, and enforcement must be integrated with other security functions. This paper examines these issues, and more specifically reviews the modifications required to the TIS http-gw proxy to support Internet content regulation.

1 Introduction

On the 26th May, 1999, the *Broadcasting Services Amendment (Online Services) Bill*, that amends the *Broadcasting Services Act of 1992* was passed by the Australian Senate [4]. The ABA was commissioned to investigate the issue of regulating the Internet. This Bill was the result of work undertaken by the Australian Broadcasting Authority (ABA) since July 1995.

Under the Bill, the ABA becomes the first point of contact for people wishing to make a complaint about online content which they feel is offensive. The ABA's complaints handling role commenced on the 1st January 2000, and the ABA has the authority to judge whether material is suitable or should be prohibited (see Appendix A). Internet Service Providers (ISPs) must have access to a list of prohibited sites, and must filter the prohibited sites by 6pm on the next business day after the sites have been listed [4].

There are many interesting moral and practical issues associated with Internet censorship. However we leave these for others to consider, and in this paper

we concentrate on the implementation issues for an ISP to control content. Controlling content is a security issue: the ISP will need to be sure that it can accurately obtain the ABA’s current prohibited list, and the control of content will need to be performed in a secure fashion, most likely by the modification of an existing security component.

This paper examines both topics. Section 2 examines the issues relating to obtaining a list from the ABA that can be checked for authenticity. This can be achieved by various reasonable methods available to date. Section 3 will examine the placement and implementation of the filtering required by the ISP. The ISP’s firewall Hypertext Transfer Protocol (HTTP) proxy seems the most sensible position for this filtering and the paper examines modification to the Trusted Information Systems (TIS) [3] http-gw proxy to support this filtering. Furthermore, we present a more efficient configuration of using the TIS http-gw proxy combined with Squid cache proxy [6] in Section 4. The paper finishes with our conclusions (Section 5).

2 Authenticity of the Prohibited Site List

The ABA will have the responsibility of maintaining a *prohibited site list*. This list should contain the Uniform Resource Locators (URLs) and/or IP addresses of sites deemed to be prohibited. Table 1 gives an example of such a list.

Table 1. An example of prohibited site list.

198.161.4.33
http://www.xrated.com
http://www.xxx.net
http://www.xtreme.org
ftp://zigzag.com.au
http://www.nuclearbomb.org
http://www.darkeyes.com
121.40.5.83
192.3.131.6
http://www.xtra-xray.com
http://www.homemadegun.net

ISPs have the responsibility of downloading the prohibited site list at regular intervals and using this to filter the client requests that it receives.

One issue that the ABA must address here is ensuring the authenticity of the list. That is, the ISP must be sure that the list it has downloaded from the ABA has not had unauthorised modifications either at the ABA or during download since the list was created. The ISP must also be sure of the author of the list. Confidentiality of the list is not required.

For ensuring the authentic prohibited site list to be downloaded, some methods might be employed, such as using Secure Socket Layer (SSL) [1] connection or to have the ABA digitally sign it using a suitable public key algorithm (for example, RSA, DSS or an elliptic curve scheme) to download. The ABA might use PGP [5] to digitally sign the list and let the ISPs use ftp to download such list.

If such methods for obtaining the prohibited site list are not preferred, there is another way to obtain the list, the ABA periodically sends a digitally signed list by email to ISPs. This will allow both the author of the list to be verified and also the integrity of the list to be checked. The signature verification code will need to be built into software at the ISP. Of course special attention must be paid to ensuring the authenticity of the regulatory body's public key (in the absence of a suitable public key infrastructure (PKI)).

3 Filtering of Prohibited Sites

This section discusses the placement and implementation issues associated with filtering prohibited sites.

3.1 Positioning of the Filter

The ISPs may employ an HTTP proxy filter (for example, the TIS http-gw proxy) or an HTTP proxy cache (for example, the Squid cache proxy) or even a combination of the two to filter and to cache client HTTP requests. The combination may allow ISPs to control, monitor and cache such requests.

If an HTTP proxy filter is employed alone, a typical configuration is shown in Figure 1. Client browsers are configured to use the proxy, and all requests are made to the proxy, which in turn makes requests to the actual Internet web server, and replies are fed back through the proxy to the browser.

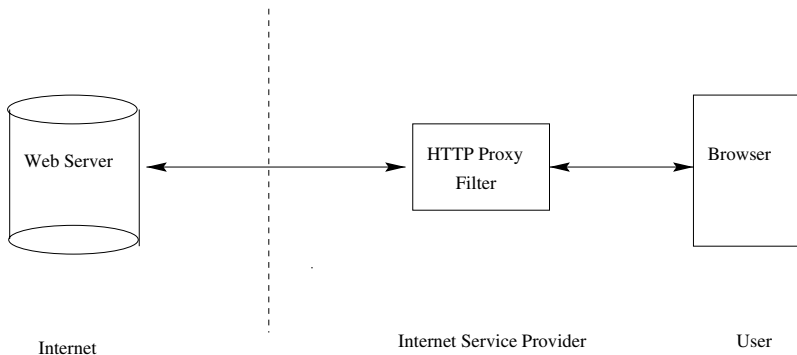


Fig. 1. HTTP proxy filter for Web Requests

Traditionally, the reason for using proxies has been to reduce the bandwidth requirements and to improve performance. Caching proxies are commonly used to try to minimise network traffic by storing copies of recently downloaded HTTP content so that subsequent requests for the same content do not result in the web server responding with the same content repeatedly. While some caching proxies have filtering built in, it is usually acknowledged that the cache is not designed to filter traffic efficiently. It is suggested that the most suitable arrangement is to have a light-weight (i.e., efficient) filtering proxy placed between the client browser and the caching proxy. This prevents requests destined for banned sites from reaching the caching proxy. The possible arrangement of employing both HTTP proxy filter and HTTP proxy cache is shown in Figure 2.

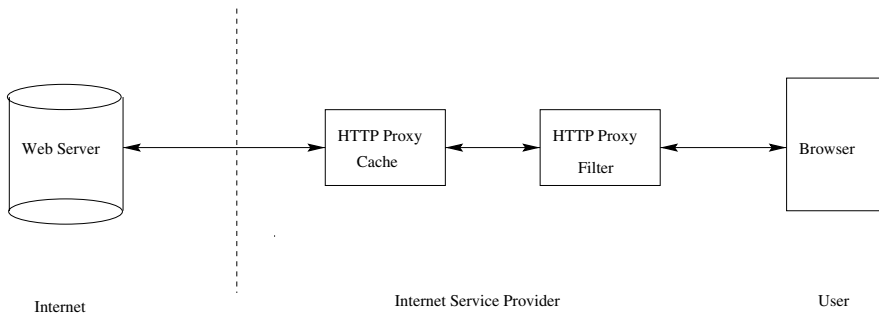


Fig. 2. An HTTP proxy cache is employed in the firewall HTTP proxy

HTTP proxies are commonly supplied as part of the ISP’s firewall system. Virtually every known application-level firewall provides an HTTP proxy. As it is anticipated that the firewall will store hundreds or even thousands of prohibited sites, to implement the prohibited site list filtering ISPs may need to modify, or have their suppliers modify, their firewall’s HTTP proxy.

3.2 Three-Party HTTP Proxy

Standard HTTP proxies are two-party. That is, they communicate with the browser (the first party) and the web server (the second party). To implement Internet content regulation the HTTP proxy will need to be extended to include a third party: the proxy will need to be modified to communicate with the ABA site to download the prohibited site list. Figure 3 below describes how the proxy system must have the capability of contacting and downloading such list at a specific site (say, the ABA site) and in a specific period (say, every 6 pm) on a daily basis.

Apart from ensuring that the list has been properly authenticated at the time it is downloaded, it is vital that the integrity of the list be maintained at

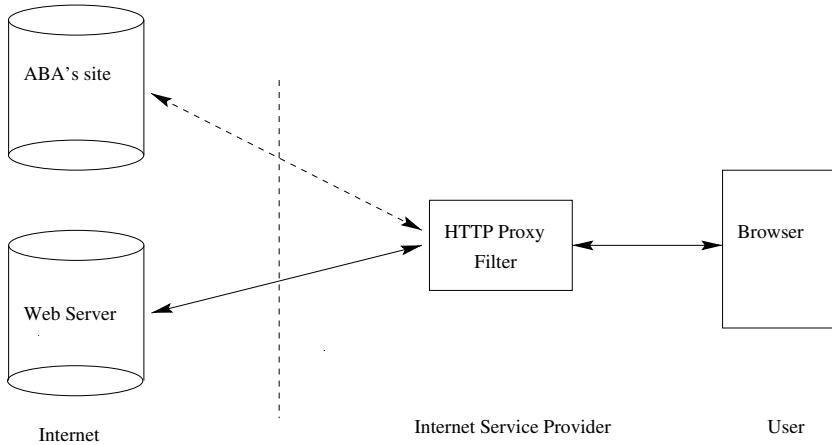


Fig. 3. A Three-Party HTTP Proxy is employed contacting ABA's site periodically

all stages throughout its life. One possible tool for achieving this is a file system integrity checker such as Tripwire [2].

4 Modification to the TIS Firewall Toolkit HTTP Proxy

A major issue to address in developing a proxy for filtering HTTP traffic based on a destination URL is the performance of the proxy. If the size of the prohibited site list is large then a filtering proxy may introduce a significant delay making the elapsed time of an HTTP request and the corresponding response prohibitive.

In order to investigate the delay introduced by URL filtering we have modified the source code of the `http-gw` proxy which forms part of the Trusted Information Systems (TIS) Firewall Toolkit (FWTK). The source code for the FWTK is freely available from the Internet [3].

The TIS FWTK uses a global configuration file which is usually called `netperm-table`. TIS's `http-gw` includes a rudimentary filtering mechanism which allows the banning of certain destinations using `-dest !this.site.com` (for example). This mechanism does not scale well and is in fact limited to only a small number of banned sites since they must be listed in a single line which is limited to only 1024 characters.

In this modification, a new separate text file is created. This new file is simply located in the same directory as `netperm-table`. We chose to put the banned URL list in a file called `ban-table`¹. Ideally this file should be read less frequently to reduce the overhead associated with processing the list once a connection is made. It is proposed that an ISP would download this file once a day from the ABA.

¹ Of course the name of this file could be configurable through the `netperm-table`.

The ABA may put such sites in the form of URLs or IP addresses in a random manner in a text file, one site in every line. At the ISP's side, the downloaded list is then verified and is put in the **ban-table** mentioned above in the same directory as the **netperm-table**. This table is then called from this **netperm-table** configuration file when the TIS http-gw is run at first time. In this http-gw modification, the proxy reads the downloaded file and put it in a binary tree in memory in order to gaining a faster lookup for the requested site. The http-gw proxy will be re-run again when there is a new valid prohibited site list downloaded. A **cron** job utility in Unix machine may be used to do this task stated in a script file.

A number of options are available in performing the processing of the prohibited site list. We have implemented the following configurations and present a comparison of each of the techniques from a performance perspective.

1. A list stored in the **netperm-table**.
2. A flat list read sequentially from a file.
3. A list stored sequentially in memory as a linked list (the file is read only once for multiple connections to the proxy).
4. A list stored in memory as a binary tree (once again, the file is read only once at which time the binary tree is constructed).

The first of these options was discussed briefly above. The http-gw provides a mechanism for banning certain destination sites but is limited in the number of sites that can be stored as being previously described. The second mechanism trialled was to read through the file sequentially looking for a match between the URLs stored in the **ban-table** and the URL in the HTTP request. A natural progression from the second choice was to store the list in memory so that file access times (typically slow) did not impact so greatly on the search time. In this case the (memory-resident) list was searched sequentially. The fourth choice was designed to streamline the search of the memory-resident list. This was achieved by storing the list in a binary tree. It should be noted that a binary tree is more likely to be *balanced* if the **ban-table** is ordered randomly. A sorted **ban-table** will produce a binary tree equivalent to a sequential linked list (an undesirable situation). We have put, in the experiment, the "target" site at the end of the **ban-table** in order that this site is located as deeply as possible in the binary-tree. Hence, if the ABA is likely to providing a sorted prohibited site list, the ISPs must then randomise it first to increase the average search performance.

Table 2 gives a comparison of each of these techniques. It can be seen that the speed-up associated with loading the list into memory and processing it there, rather than from disk is significant, although not astonishing (roughly twice as fast). The most significant improvement is achieved through processing the list when it has been stored in a binary tree - the improvement is almost two orders of magnitude for a list containing one million entries². These results were expected and serve to highlight the importance of suitable data structures in situations where computational efficiency are vital. All results are in milliseconds (ms).

² The entries were created randomly and hence the tree is considered well-balanced.

Table 2. Average timing results comparison (milliseconds)

Number (of URLs)	http-gw (FWTK)	read file (sequential)	linked list (in memory)	binary tree (in memory)
1	9.8	9.8	9.6	9.9
10	17.6	9.9	10.0	9.9
20	26.4	10.8	10.5	9.9
30	32.8	9.8	10.1	9.8
40	41.8	10.4	10.1	7.2
50	50.0	10.4	9.5	9.9
57	55.0	NA	NA	NA
100	NA	10.0	9.8	9.9
1,000	NA	11.3	10.7	10.5
10,000	NA	26.2	18.4	11.3
100,000	NA	310.8	195.6	11.6
500,000	NA	1,025.1	594.3	13.3
750,000	NA	1,517.7	800.6	12.9
1,000,000	NA	1,910.6	984.5	13.3

Besides the elapsed time to search within such list, we must also consider that if a caching proxy is used between the Internet and http-gw proxy, the delay will only be experienced on the first access to any particular site. A caching proxy is therefore always recommended.

Meanwhile, when the prohibited site list is large it becomes unacceptable to read the list into memory more frequently than is necessary. To overcome this it is necessary to utilise *shared memory* so that different (forked) instantiations of the proxy application can each access the memory containing the list, concurrently. A single memory-resident copy of the list would also facilitate easier update of the list. It is noted that shared memory techniques are well established and, hence, do not fall within the scope of this work.

5 Conclusions

Modification of the public domain code is a straight forward and reasonably cheap method for obtaining an HTTP proxy applicable to Internet content regulation. The modification was successfully done extending from a small amount of banned sites to a very large amount of banned sites. The binary-tree method employed in the http-gw proxy is very helpful to short the elapsed time scanning up to a million sites.

The two security components in the implementation of Internet content regulation: verification of the prohibited site list and filtering issue which are involved here must be carefully considered. The authenticity of the prohibited site list is very important to ensure the Internet content regulation works well and attempts to reduce the chance of bypassing such filtering must be done, such as

using auto-lookup for Domain Name Service to its IP address or vice versa and also its aliases must be recognised.

The modified proxy might need to be investigated further to monitor its performance when the proxy serves from tens up to hundreds of clients. An auto-lookup facility for DNS name and its related IP address also seems to be useful for future work.

References

1. Netscape Communication Corporation. Introduction to SSL. <http://developer.netscape.com/docs/manuals/security/sslin>, 5 February 2000.
2. Eugene H. Spafford Gene H. Kim. Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection. *Purdue Technical Report*, 5 February 2000.
3. Trusted Information Systems. Internet Firewall Toolkit. <http://www.tis.com/research/software>, 2 June 1999.
4. Internet Industry Association website. Summary of the Main Features of the Broadcasting Services Amendment (On-line Services) Bill 1999. <http://www.iiia.net.au/news/990611.html>, 18 February 1999.
5. MIT website. Integrating PGP with Various E-Mail Products. <http://web.mit.edu/network/pgp.html>, 5 February 2000.
6. Squid website. Squid Web Proxy Cache. <http://squid.nlanr.net>, 9 August 1999.

A When a Site is Put into the Prohibited Site List

In its implementation, the ABA lists sites which contain material containing detail instruction in crime, violence or drug use, child pornography, bestiality, excessively violent or sexually violent material, or contain real depiction of actual sexual activity. Such contents are classified into 'RC' or 'X' by the Classification Board.

Meanwhile, such sites which rated 'R' is not subject to a restricted access system may contain material with excessive and/or strong violence or sexual violence, implied or simulated sexual activity, or material containing depictions of things which require assistance of an adult. For these sites, the ABA will put the sites into the list if there is a complaint from an Australian resident, a body corporate in Australia, or the Commonwealth Government or a State or a Territory.

Cryptanalysis of the m – Permutation Protection Schemes

Hongjun Wu, Feng Bao, Dingfeng Ye, and Robert H. Deng

Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613

{hongjun, baofeng, dfye, deng}@krdl.org.sg

Abstract. Anderson and Kuhn have proposed the EEPROM modification attack to recover the secret key stored in the EEPROM. At ACISP98, Fung and Gray proposed an m – permutation protection scheme against the EEPROM modification attack. At ACISP99, Fung and Gray pointed out that in their original scheme, a secret key with too small or too large Hamming weight could be recovered easily. Then they proposed a revised m – permutation protection scheme and claimed that their revised scheme does not leak any information of the secret key. In this paper, we break completely both the original and the revised m – permutation protection schemes. The original scheme is broken with about $2\log_2 n$ devices from the same batch and about $(3\log_2 n + 2) \times m \times n$ probes (n is the length of the secret key and m is the amount of permutations). The revised m – permutation protection scheme is more vulnerable than the original one. It could be broken with only one device and about $m \times n^3 / 3$ probes.

1 Introduction

The design of tamperproof device is an important issue in the applications of cryptographic systems. There are basically two types of attacks against the tamperproof devices. The direct attack is to reverse engineer the device with advanced hardware technology. Another type of attacks is to force the device to produce computational errors. Boneh, DeMillo and Lipton have developed such an attack against tamperproof device [4]. In their attack random errors are introduced into the data on the device. The random errors cause a corresponding erroneous output that can be used to deduce the key. This attack is simple but powerful and is able to break the devices using RSA. A similar attack was reported independently by Bao, Deng et. al. who showed how to attack the RSA, El Gamal and Schnorr Signature schemes [2]. Biham and Shamir later introduced the Differential Fault Analysis or DFA [3]. DFA can be applied to recover a block cipher key from a sealed tamperproof device. To resist these fault-related attacks the device needs to perform fault checking before outputting the encrypted (or decrypted, signed) result.

Anderson and Kuhn introduced the EEPROM modification attack that is quite general and practical. In their attack, an attacker is assumed to be able to write

arbitrary values to arbitrary locations of the EEPROM, where the secret key is stored, but cannot read a value from the EEPROM. This is because the cost of writing a value to EEPROM is much lower than that of reading a value from EEPROM, i.e., the writing can be done with the low-cost equipment, such as microprobes, while the reading requires much more expensive equipment, such as an electro-optical probe.

To protect the device against the EEPROM modification attack, Fung and Gray proposed a cascaded m -permutation scheme that uses an $(m \times n)$ -bit encoding for an n -bit key [5]. Each batch of devices employs the same permutations (i.e., encoding). The permutation wiring is secret and it is assumed that the attacker has no equipment to reveal the wiring. Fung and Gray claimed that the attack on the m -permutation scheme requires $O(n^m)$ probes to compromise the key. In [6], Fung and Gray pointed out that if the Hamming weight of a key is too small or too large, the key could be recovered easily. Then they introduced the revised scheme in which random numbers are introduced to hide the information about the Hamming weight of the secret key.

In this paper, we show that both the original and the revised schemes are not secure. For the original scheme, there exists an attack that could recover the m permutations with about $2\log_2 n$ devices from the same batch and about $(3\log_2 n + 2) \times m \times n$ probes. The m -permutation scheme achieves only linear growth of complexity with a linear growth of the amount of permutations.

In the revised scheme, m random numbers are introduced into each device to hide the information of the Hamming weight of the secret key. However, these random numbers leak the information about the secret permutations. By modifying these random numbers, we could recover the mappings between those permutations, i.e., the m -permutation scheme could be reduced to one permutation scheme. Thus with only one device, we break the revised m -permutation scheme with $m \times n^3 / 3$ probes. Since only one device is needed in this attack, we consider that the revised scheme is more vulnerable than the original scheme. We try to strengthen the revised scheme by eliminating the flaw introduced by the random numbers. However, there still exist an attack that could recover those m permutations with about $m \times n$ devices from the same batch and about $3.5(m \times n)^2$ probes.

A fairly simple and efficient scheme to defeat the EEPROM modification attack is proposed in this paper. By restricting the Hamming weight of the key to be half of n , only one permutation is needed.

This paper is organized as follows. The EEPROM modification attack is introduced in Section 2. Fung and Gray's original and revised m -permutation protection schemes are introduced in Section 3. We break the original and the revised m -permutation scheme in Section 4 and Section 5, respectively. In Section 6, we break the strengthened version of the revised scheme. Section 7 gives our simple and efficient protection scheme. Section 8 concludes the paper.

2 The EEPROM Modification Attack

In [1], Anderson and Kuhn proposed the EEPROM modification attack. It is a physical attack in which two microprobing needles are used to set or clear target bits in order to infer them. It is assumed that EEPROM bits cannot be read directly since the equipment required is much more expensive than the microprobing needles. In the EEPROM modification attack, if one bit of the secret key is set correctly, there would be no error in the output of the device; otherwise, error occurs. The secret key can thus be determined bit by bit.

Anderson and Kuhn's attack in [1] is with respect to a DES key. The more general attack described by Fung and Gray [5] is given below:

```

for  $i = 0$  to  $n-1$ 
    set the  $i^{th}$  bit to 1;
    operate the device;
    if the device gives the correct output, then conclude
        that the bit is 1; otherwise, conclude that the bit is 0
    and reset it to 0.
  
```

In addition to requiring only low-cost equipment, this attack can be carried out with very few probing actions. In particular, it takes $1.5n$ probes on the average to recover an n – bit key.

3 The m – Permutation Protection Schemes

The m – permutation schemes [5,6] provide a physical encoding (m permutations) of keys, along with a logical chip design and hiding the permutation wiring beneath the surface of the chip. The m permutations are considered as the „batch key“ which is known only to the manufacturers and to those who are legitimately programming the device. For example, the devices may be manufactured in batches of 10,000 devices all with the same batch key. A single customer purchases a batch of devices and is given the batch key so that he can program secret keys into the cards.

There are several assumptions made. Firstly, the attacker is assumed to be a „clever outsider with moderately sophisticated equipment“. Secondly, the encoded key is assumed to be stored in EEPROM and that the attacker cannot read the EEPROM directly. Finally, it is assumed that the attacker is not able to see the exact wiring (i. e., the batch key) of the devices. The following notations are used in the rest of this paper:

- K : The actual key bit vector with length of n bits. It is to be used by the card in encrypting, signing, etc.
- P : The physical key bit vector with length of p bits. It is the actual bit pattern stored in the EEPROM.
- π : A permutation function, $\pi : \{0, 1, 2, \dots, n-1\} \rightarrow \{0, 1, 2, \dots, n-1\}$

- π^{-1} : The inverse function of the permutation π
- Element of π : An element of the permutation table $(i, \pi(i))$ ($i \in \{0, 1, \dots, n-1\}$).
- $\pi(K)$: It denotes the permuted result of K under the operation of π , i.e.,
 $(\pi(K))_{\pi(i)} = (K)_i$ for $0 \leq i \leq n-1$, where $(K)_i$ denotes the value of the i^{th} bit of K .

3.1 One-Permutation Scheme [5]

In this approach, the manufacturer chooses a random permutation of the n -bit key as the batch key and works as follows:

Scheme 1. One-permutation Scheme

1. The device manufacturer chooses randomly a permutation π
2. Set $P = \pi(K)$
3. The wiring implements the inverse of π
4. The device reads K from P since $\pi^{-1}(P) = \pi^{-1}(\pi(K)) = K$.

The attacker can find the value of P as described in Section 2. Then the permutation π can be determined as follows. The attacker first sets P as a vector with Hamming weight one, then operates the device to obtain an output. The value of K can be determined since the Hamming weight of K is also one due to the fact that $K = \pi^{-1}(P)$. Thus one element of π is known. Repeat this process for $n-1$ times, the permutation π can be determined with about $3n$ probes.

3.2 m – Permutation Protection Scheme [5]

In the m – permutation scheme, the manufacturer chooses m random permutations of the n – bit key as the batch key and works as follows:

Scheme 2. m – permutation protection scheme

1. The device manufacturer chooses m permutations: $\pi_0, \pi_1, \dots, \pi_{m-1}$
 $\pi_i : \{0, 1, 2, \dots, n-1\} \rightarrow \{0, 1, 2, \dots, n-1\}$ for $0 \leq i \leq m-1$
2. Let $P = P_0 \mid P_1 \mid \dots \mid P_{m-1}$, where $\mid$ denotes concatenation and $P_i = \pi_i(K)$.
3. The wiring implements the inverse of those m permutations.
4. The device reads K_0 from P_0 , K_1 from P_1 , ..., K_{m-1} from P_{m-1} .
 If $K_0 = K_1 = \dots = K_{m-1}$ is not true, the device gives an error message.

The attack to break the one-permutation scheme could not be applied directly to the m – permutation scheme since without knowing the secret permutations, the modified values of P_i can pass the detection with negligible probability.

As pointed out by Fung and Gray, some secret key with small Hamming weight could be recovered easily [5]. To eliminate such weakness, Fung and Gray proposed the revised m – permutation scheme [6] to hide the information about the Hamming weight of the secret key by introducing m random numbers into each device. The revised scheme is given in the next subsection.

3.3 The Revised m – Permutation Protection Scheme

In the revised m – permutation scheme, m random numbers are introduced into each device.

Scheme 3. Revised m – permutation protection scheme

1. Choose m as an odd number.
2. The device manufacturer chooses m permutations $\pi_0, \pi_1, \dots, \pi_{m-1}$ as the secret information (batch key) for a batch of devices.
 $\pi_i : \{0, 1, 2, \dots, n-1\} \rightarrow \{0, 1, 2, \dots, n-1\}$ for $0 \leq i \leq m-1$
3. Randomly choose m n – bit words $K_{D_0}, K_{D_1}, \dots, K_{D_{m-1}}$ for each device.
4. Store in the device $P = P_0 | P_1 | \dots | P_{m-1}$ where

$$P_i = \pi_i(K \oplus \pi_{i+1 \bmod m}(K_{D_{i+1 \bmod m}}) \oplus \pi_{i+2 \bmod m}(K_{D_{i+2 \bmod m}}))$$
5. Store in the device $P_D = P_{D_0} | P_{D_1} | \dots | P_{D_{m-1}}$ where $P_{D_i} = K \oplus K_{D_i}$.
6. The wiring implements the inverse of those m permutations.
7. The device decodes the key as $K = \bigoplus_{i=0}^{m-1} \pi_i^{-1}(P_i)$
8. The device then computes

$$K_{and} = \bigwedge_{i=0}^{m-1} (\pi_i^{-1}(P_i) \oplus \pi_{i+1 \bmod m}(P_{D_{i+1 \bmod m}} \oplus K) \oplus \pi_{i+2 \bmod m}(P_{D_{i+2 \bmod m}} \oplus K))$$

$$K_{or} = \bigvee_{i=0}^{m-1} (\pi_i^{-1}(P_i) \oplus \pi_{i+1 \bmod m}(P_{D_{i+1 \bmod m}} \oplus K) \oplus \pi_{i+2 \bmod m}(P_{D_{i+2 \bmod m}} \oplus K))$$

where $\wedge$ and $\vee$ indicates logical *AND* and *OR* respectively.

If $K_{and} = K_{or} = K$, then the device uses K in the crypto application; else return an error message.

In the revised scheme (Scheme 3), the Hamming weight of the secret key is unknown after P and P_D being recovered under the EEPROM modification attack. Fung and Gray claimed that the attacker has only a probability of $2^{-(n-1)}$ to guess the n – bit key

K since the m permutations are unknown to the attacker. However, as we will present in Section 5, there exists an attack that can recover those permutations with only one device and about $m \times n^3 / 3$ probes. Once those permutations are recovered, the whole batch of devices is broken and the secret keys can be determined easily.

4 Cryptanalysis of the Original m – Permutation Protection Scheme

Fung and Gray have pointed out a weakness in their original m – permutation protection scheme that some keys with small or large Hamming weight could be recovered easily. In this section, we present an attack to break completely the original scheme with about $2 \log_2 n$ devices from the same batch and about $3 \times m \times n \times \log_2 n$ probes. Our attack consists of two steps. In the first step, we determine the mappings between those m permutations by analyzing $2 \log_2 n$ devices, i.e., we reduce the m – protection scheme to one-permutation scheme. In the second step, we recover the remaining permutation. We start with the first step.

Assume that about $2 \log_2 n$ devices from the same batch are available and the values of P ($P = P_0 | P_1 | \dots | P_{m-1}$) in these devices are determined already by applying the EEPROM modification attack. The amount of probes needed here is about $3 \times m \times n \times \log_2 n$. We denote P_j^i as the value of P_j in the i th device.

We know that $P_j^i = \pi_j \circ \pi_0^{-1}(P_0^i)$ since $P_j^i = \pi_j(K^i)$ and $P_0^i = \pi_0(K^i)$ (K^i is the secret key in the i th device). The permutation $\pi_j \circ \pi_0^{-1}$ is determined as follows. Consider two $(2 \log_2 n) \times n$ binary matrices M and N with the i th row $M^i = P_j^i$ and $N^i = P_0^i$. We note that $M^i = \pi_j \circ \pi_0^{-1}(N^i)$, i.e., M is obtained by exchanging the columns of N under the permutation $\pi_j \circ \pi_0^{-1}$. Clearly, if all the columns of N are different, the permutation $\pi_j \circ \pi_0^{-1}$ can be determined uniquely. Assume that all the keys are randomly generated, then the columns of the matrix N are n random elements in a set with n^2 elements. From the birthday paradox, the probability that all these elements are different is about 0.61 (for almost all the key length $40 \leq n \leq 4096$). Thus the permutations $\pi_j \circ \pi_0^{-1}$ ($0 < j \leq m-1$) could be uniquely determined with about $2 \log_2 n$ devices with probability about 0.61. If a few elements of the permutations $\pi_j \circ \pi_0^{-1}$ ($0 < j \leq m-1$) could not be recovered (i.e., some columns of the matrix N are with the same value), some key bits would be unknown. However, those key bits can be determined easily by exhaustive search. In the rest of this section, we simply assume that $\pi_j \circ \pi_0^{-1}$ ($0 < j \leq m-1$) are uniquely determined already. Then there is only one unknown permutation left, i.e., if we can find π_0 , we will know all the π_i since $\pi_i = (\pi_i \circ \pi_0^{-1}) \circ \pi_0$. We give below the details to recover π_0 .

To recover π_0 , we need to write a key with Hamming weight one into the device correctly so that $\pi_0^{-1}(P_0) = \pi_1^{-1}(P_1) = \dots = \pi_{m-1}^{-1}(P_{m-1})$. If P_0 is set as an n – bit word with Hamming weight one, then P_i could be determined easily since $P_i = \pi_i \circ \pi_0^{-1}(P_0)$ and $\pi_i \circ \pi_0^{-1}$ is known already. Thus we are able to write any key with Hamming weight one into the device. Once knowing the device output, the value of the key with Hamming weight one could be determined. Since $P_0 = \pi_0(K)$, one element of π_0 is determined. Set the bit with value 1 at different positions in P_0 and repeat the attack, we could recover π_0 with about $2 \times m \times n$ probes.

From π_0 and $\pi_i \circ \pi_0^{-1}$ ($i = 1, \dots, m-1$), we know all the permutations and thus can break the m – protection scheme. About $2 \log_2 n$ devices are needed in this attack and the total amount of probes needed is about $3m \times n \times \log_2 n + 2m \times n = (3 \log_2 n + 2) \times m \times n$.

The attack in this section needs about $2 \log_2 n$ devices from the same batch. In case devices from a number of batches are well mixed, a simple method could be used to group those devices. We write the P of one device into all the devices, then those devices that operate properly belong to the same batch.

5 Cryptanalysis of the Revised m – Permutation Protection Scheme

In the revised scheme, m random numbers are introduced into each device to hide the information of the Hamming weight of the secret key. However, the revised scheme is in fact more vulnerable than the original one since those random numbers leak the information about the permutations. With only one device, those permutations could be recovered with about $m \times n^3 / 3$ probes. Similar to the attack in Section 4, the attack in this section consists of two steps. The first step of the attack is to reduce the m – protection scheme to one-permutation scheme by modifying the random numbers. The second step is to recover the remaining permutation. We start with the first step.

Assume that the values of P and P_d in a device are determined already by applying the EEPROM modification attack. We note that those m numbers K_{d_i} ($i = 0, 1, \dots, m-1$) in the revised scheme are randomly chosen. Obviously, if we replace any K_{d_i} with another random number, the value of the secret key K will not be affected and the device will operate properly. Suppose we want to modify the j th bit of a particular random number K_{d_i} . This bit is denoted as $K_{d_i,j}$. We know that this bit appears only in P_{d_i} , $P_{i-1 \bmod m}$ and $P_{i-2 \bmod m}$ since

$$P_{d_i} = K \oplus K_{d_i}, \quad (1)$$

$$P_{i-1 \bmod m} = \pi_{i-1 \bmod m}(K \oplus \pi_i(K_{D_i}) \oplus \pi_{i+1 \bmod m}(K_{D_{i+1 \bmod m}})) \quad (2)$$

$$P_{i-2 \bmod m} = \pi_{i-2 \bmod m}(K \oplus \pi_{i-1 \bmod m}(K_{D_{i-1 \bmod m}}) \oplus \pi_i(K_{D_i})) \quad (3)$$

Modifying the bit $K_{D_{i,j}}$ in P_{D_i} is trivial since we only need to invert the value of $P_{D_{i,j}}$. Modifying $K_{D_{i,j}}$ in $P_{i-1 \bmod m}$ and $P_{i-2 \bmod m}$ without knowing the permutation $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$ requires about $n^2/2$ trials on average. Thus with about n^2 probes, we could modify the value of $K_{D_{i,j}}$ successfully (If it is not modified correctly, the device gives error message). If the values of $P_{D_{i,j}}$, $P_{i-1 \bmod m, j'}$ and $P_{i-2 \bmod m, j''}$ are modified and the device operates properly, we know from (1), (2) and (3) that

$$\pi_{i-1 \bmod m} \circ \pi_i(j) = j', \quad \pi_{i-2 \bmod m} \circ \pi_i(j) = j''$$

We thus determined one element of $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$. Repeat this attack for the rest bits of K_{D_i} , $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$ are recovered with about $\sum_{i=2}^n i^2$ probes. Similar attack can be applied to recover the permutations $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$ ($i=0,1,\dots,m-1$). From these permutations, $\pi_i \circ \pi_0^{-1}$ ($i=1,\dots,m-1$) are obtained as follows:

$$\begin{aligned} \pi_i \circ \pi_0^{-1} &= (\pi_i \circ \pi_{i-1}^{-1}) \circ (\pi_{i-1} \circ \pi_{i-2}^{-1}) \cdots (\pi_1 \circ \pi_0^{-1}) \\ &= (\pi_i \circ \pi_{i+1 \bmod m}^{-1} \circ \pi_{i+1 \bmod m}^{-1} \circ \pi_{i-1}^{-1}) \circ (\pi_{i-1} \circ \pi_i^{-1} \circ \pi_{i-2}^{-1}) \\ &\quad \cdots (\pi_1 \circ \pi_2 \circ \pi_2^{-1} \circ \pi_0^{-1}) \\ &= (\pi_i \circ \pi_{i+1 \bmod m}) \circ (\pi_{i-1} \circ \pi_{i+1 \bmod m})^{-1} \circ (\pi_{i-1} \circ \pi_i) \circ (\pi_{i-2} \circ \pi_i)^{-1} \\ &\quad \cdots (\pi_1 \circ \pi_2) \circ (\pi_0 \circ \pi_2)^{-1} \end{aligned} \quad (4)$$

After $\pi_i \circ \pi_0^{-1}$ ($i=1,\dots,m-1$) being recovered, only π_0 remains to be recovered. To recover π_0 , we need to write a key with Hamming weight one into the device correctly, i.e., the values of K and K_{D_i} ($i=0,1,\dots,m-1$) should be set correctly in P and P_D . We deal first with K . We choose K as an n -bit word with Hamming weight one. It appears in P_i ($i=0,1,\dots,m-1$) and P_{D_i} ($i=0,1,\dots,m-1$). The value of K in P_{D_i} is K itself since $P_{D_i} = K \oplus K_{D_i}$ ($i=0,1,\dots,m-1$). But K_0 , the value of K in P_0 , is unknown since $K_0 = \pi_0(K)$ and π_0 is unknown. We randomly set K_0 as an n -bit word with Hamming weight one. The probability that $K_0 = \pi_0(K)$ is n^{-1} . After setting the value of K in P_0 , the value of K in P_i could be determined since $K_i = \pi_i(K) = \pi_i \circ \pi_0^{-1}(K_0)$ and $\pi_i \circ \pi_0^{-1}$ is known from (4). Thus the values of K in P and P_D are determined with probability n^{-1} . We then deal with the random

numbers $K_{D_i} (i = 0, 1, \dots, m-1)$. The simplest way is to set their values as zero. Then their values in P and P_D are zero. Now, we are able to write a key with Hamming weight one into a device with success rate n^{-1} . Once the device operates properly, we know that the key is written successfully into the device. If that happens and the bit K^j and K_0^j are with value one, then $j' = \pi_0(j)$, i.e., one element of π_0 is recovered. The amount of probes needed is about $m \times n$. Repeat this attack, we could finally recover π_0 with about $\sum_{i=2}^n m \times i$ probes.

After recovering π_0 and $\pi_i \circ \pi_0^{-1} (i = 1, \dots, m-1)$, we break the revised m – protection scheme completely. Only one device is needed in this attack and the total amount of probes needed is about $\sum_{i=2}^n m \times i^2 + \sum_{i=2}^n m \times i \approx m \times n^3 / 3$.

In the next section, we will discuss whether the revised protection scheme could be strengthened or not. Our analysis gives negative result.

6 Is It Possible to Strengthen the Revised Scheme

The attack in Section 5 is based essentially on the fact that each random number appears at only three locations in the EEPROM. The mappings between the permutation tables could be determined by modifying one by one the bit of those random numbers. To resist the attack in Section 5, each random number should appear at far more than three locations in the EEPROM. For example, the P_i in the revised scheme can be modified as

$$P_i = \pi_i(K \oplus \pi_{i+1 \bmod m}(K_{D_{i+1 \bmod m}}) \oplus \pi_{i+2 \bmod m}(K_{D_{i+2 \bmod m}}) \cdots \oplus \pi_{i+8 \bmod m}(K_{D_{i+8 \bmod m}})),$$

then each random number appears at eight positions in the EEPROM. However, the revised scheme strengthened in this way is still not secure. We can recover those permutation tables if about $m \times n$ devices from the same batch are available. In the rest of this section, we present a new attack to break only Scheme 3, but the same attack can be applied to the scheme where each random number appears at a number of locations in the EEPROM.

Assume that about $m \times n$ devices from the same batch are available and the values of P ($P = P_0 | P_1 | \dots | P_{m-1}$) and P_D ($P_D = P_{D_0} | P_{D_1} | \dots | P_{D_{m-1}}$) in these devices are determined already by applying the EEPROM modification attack. The amount of probes needed here is about $3 \times (mn)^2$. We denote P^i and P_D^i as the values of P and P_D in the i th device, respectively.

Our aim is to write a P' with Hamming weight one into the device. Fung and Gray have considered this kind of attack and concluded that it is impossible to apply it to break their revised scheme [6]. They consider that if a P' with Hamming weight one is written into the EEPROM and the value of P_D' is randomly set, then the probability

that the device could operate properly (no error message) is negligibly small (about 2^{-mn}). However, with about $m \times n$ devices from the same batch, it is possible to construct a right pair (P', P'_d) (a right pair means that with which the device operates properly and gives no error message). The method to construct such a pair is given below.

Algorithm 1. This algorithm is to construct a pair (P', P'_d) for any given P' . It needs $m \times n$ devices from the same batch.

1. Form two $mn \times mn$ binary matrices M and N with the i th column $M_i = (P^i)^T$ and $N_i = (P^i_d)^T$.
2. Solve the linear equations $M \cdot x^T = P'^T$. Let $P'_d = (N \cdot x^T)^T$.
3. The pair (P', P'_d) is the one we need.

Then we need to show: 1) the equation $M \cdot x^T = P'^T$ could be solved, i.e., the matrix M is invertible with large probability, 2) with the pair (P', P'_d) generated in Algorithm 1, the device operates properly.

To show that the Matrix M is invertible with large probability, we start with the following theorem.

Theorem 1. In Scheme 3 (the revised protection scheme), assume that all the keys and random numbers in the devices are generated independently and randomly. Choose $m \times n$ devices from the same batch. Form an $mn \times mn$ binary matrix M , with the i th column $M_i = (P^i)^T$. Then the matrix M is a random matrix.

The proof of Theorem 1 is given in the Appendix. In theorem 1, we deal only with Scheme3. But the same result could be obtained if each random number appears at more than three locations in the EEPROM.

From Theorem 1, we know matrix M is randomly generated. So it is invertible with probability about 0.29. With slightly more than $m \times n$ devices, an invertible matrix M could be formed. So the pair (P', P'_d) in Algorithm 1 can be obtained.

Then we show below that with the pair (P', P'_d) generated in Algorithm 1, the device operates properly.

Theorem 2. In Scheme 3, choose any n devices from the same batch. Let $P' = \bigoplus_{i=0}^{n-1} P^i$ and $P'_d = \bigoplus_{i=0}^{n-1} P^i_d$. If P' and P'_d are written into the device, the device will operate properly (no error message is given).

Proof. Since $P' = \bigoplus_{i=0}^{n-1} P^i$ and $P'_D = \bigoplus_{i=0}^{n-1} P_D^i$, it is equivalent to encode a key $K' = \bigoplus_{i=0}^{n-1} K^i$ with m random numbers $K'_{D_j} = \bigoplus_{i=0}^{n-1} K_{D_j}^i$ ($j = 0, 1, \dots, m-1$). Thus, the key K' will be decoded correctly and the device will operate properly.

From the discussion above, we know that from slightly more than $m \times n$ devices, a right pair (P', P'_D) in which the Hamming weight of P' is one could be obtained easily. Once we obtained such a pair, we could recover one element of those permutation tables as follows. Suppose only the bit $P'_{i,j}$ in P' is with value one.

From Scheme 3, we know that the key K' is decoded as $K' = \bigoplus_{l=0}^{m-1} \pi_l^{-1}(P'_l) = \pi_i^{-1}(P'_i)$.

So the Hamming weight of K' is only one. By analyzing the output of the device, the value of K' can be determined. Suppose the bit with value one in K' is $K'_{j'}$, then $j' = \pi_i^{-1}(j)$, i.e., one element of π_i is recovered. Set the non-zero bit at other positions in P' and repeat this attack, we can finally recover all the permutation tables. The amount of probes needed is about $mn \times (0.5mn) = 0.5 \times (m \times n)^2$.

The attack in this section thus break the revised protection scheme even if the scheme allows each random number appearing at more than three locations. It needs slightly more than $m \times n$ devices from the same batch. The total amount of probes needed is about $3 \times (mn)^2 + 0.5 \times (mn)^2 = 3.5 \times (mn)^2$.

7 How to Prevent the EEPROM Modification Attack

We now know that all the m – permutation schemes are not secure. The flaw in those schemes is that those m permutations could be reduced to one permutation. We note that all the attacks in this paper have one common step: a key with Hamming weight one is written into the EEPROM to recover the permutation table element by element. To hide the permutation, we believe that the most essential way is to disallow a key with too small (or too large) Hamming weight being written into the device. Based on this observation, we give below a fairly simple and efficient scheme to resist the EEPROM modification attack.

Scheme 4. This scheme protects an n – bit secret key against the EEPROM modification attack with the use of only one n – bit permutation. It is the strengthened version of the one-permutation scheme given in Subsection 3.1.

1. Choose a permutation π as the batch key.
2. Choose the secret key K with Hamming weight $n/2$.
3. Let $P = \pi(K)$ and write P into the EEPROM.
4. The wiring implements the inverse of π .

5. The device reads K from P . If the Hamming weight of K is $n/2$, K is accepted; otherwise, the device gives error message.

The value of P could be recovered by applying the EEPROM modification attack. However the permutation π could not be recovered. The reason is that the output of π (the value of P) is known, but the input of π (the value of K) is unknown. By applying the EEPROM modification attack, the secret key could only be recovered by exhaustive search and the complexity is $0.5 \times \binom{n}{n/2}$. For $n=128$, the complexity is

about $2^{122.17}$ and it is sufficient to defeat the exhaustive key search. We thus believe that Scheme 4 is sufficient to resist the EEPROM modification attack. However, it should be noted that any compromise of the secret key degrades the security of the devices of the same batch.

As pointed out by the anonymous referee, some public key cryptosystems such as RSA do not allow the Hamming weight to be controlled. Scheme 4 could not be used to protect the private keys in these cryptosystems. To resist the EEPROM modification attack, we recommend the use of a hash function. The one-permutation scheme in Subsection 3.1 is used to protect the key together with its hashed value. The device hashes the key and compares the result with the hashed value stored in the EEPROM. If these two values are equal, the key is used in the crypto applications; otherwise, the device gives error message.

8 Conclusion

In this paper, we showed that Fung and Gray's original and revised m -permutation schemes are not secure. We then proposed a very simple and efficient scheme to resist the EEPROM modification attack by allowing only the key with Hamming weight $n/2$ being written into the device.

Acknowledgement

We would like to thank the anonymous referees of ACISP 2000 for their helpful comments.

References

1. R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices", in *Security protocols: International Workshop'97*, LNCS 1361, Springer-Verlag, pp.125-136, 1997.

2. F. Bao, R.H. Deng, Y. Han, A. Jeng, A.D. Narasimhaly and T. Ngair, "Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults", in *Security Protocols: International Workshop'97*, LNCS 1361, Springer-Verlag, 1997.
3. E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems", in *Advances in Cryptology - Crypto'97*, LNCS 1294, Springer-Verlag, pp. 513-525, 1997.
4. D. Boneh, R.A. Demillo and R.J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults", in *Advances in Cryptology - Eurocrypt'97*, LNCS 1233, Springer-Verlag, pp. 37-51, 1997.
5. W.W. Fung and J.W. Gray, "Protection Against EEPROM Modification Attacks", in *Information Security and Privacy - Proc. of ACISP'98*, LNCS 1438, Springer-Verlag, pp. 250-260, 1998.
6. W.W. Fung and J.W. Gray, "On m -permutation Protection Scheme Against Modification Attack", in *Information Security and Privacy - Proc. of ACISP'99*, LNCS 1587, Springer-Verlag, pp. 77-87, 1999.

Appendix. Proof of Theorem 1

Lemma 1. Consider the operation over $GF(2)$. Let N_1 and N_2 be two $m \times n$ ($m \leq n$) binary matrices. If N_1 is with rank m and each element of N_2 is generated independently and randomly, then $N_1 \cdot (N_2)^T$ is randomly generated.

Proof. Select m linearly independent columns from N_1 and form an $m \times m$ matrix N'_1 . The remaining columns form a matrix N''_1 . Select any m columns from N_2 and form an $m \times m$ matrix N'_2 . The remaining columns form a matrix N''_2 . Clearly, the matrix $N'_1 \cdot (N'_2)^T$ is randomly generated since N'_1 is an invertible matrix and N'_2 is a random matrix. The matrix $N_1 \cdot (N_2)^T = N'_1 \cdot (N'_2)^T + N''_1 \cdot (N''_2)^T$, where $N'_1 \cdot (N'_2)^T$ and $N''_1 \cdot (N''_2)^T$ are two independent matrices since N'_2 and N''_2 are independent from each other. So the matrix $N_1 \cdot (N_2)^T$ is randomly generated.

Lemma 2. An $m \times (m+1)$ binary matrix M , if $M_{i,i} = M_{i,i+1} = 1$ for $0 \leq i \leq m-2$, $M_{m-1,m} = 1$ and all the other elements are with value 0, i.e., M is in the following form:

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \quad (\text{A.1})$$

Then the rank of M is m .

Proof. Consider the last m columns of M . They form an $m \times m$ triangular matrix M' . $\det(M') = \prod_{i=0}^{m-1} M'_{i,i} = 1$, i.e., the rank of M' is m . So the rank of M is m .

Lemma 3. For an $m \times m$ binary matrix M , if $M_{i,j} = 1$ for $\{(i, j) \mid i = j, \text{ or } j = m \text{ or } i = m\}$ and the other elements are with value 0, i.e., M is in the following form:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 1 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 1 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \quad (\text{A.2})$$

Then the rank of M is m if and only if m is an odd number.

Proof. Denote an $i \times i$ matrix M in the form of (A.2) as $M_{i \times i}$. The following relationship holds:

$$\det(M_{(i+1) \times (i+1)}) = 1 + \det(M_{i \times i}).$$

Since $\det(M_{1 \times 1}) = 1$, we know that $\det(M) = m \bmod 2$. So if and only if m is an odd number, $\det(M) = 1$, i.e., the rank of M is m .

Theorem 1. In Scheme 3 (the revised protection scheme), assume that all the keys and random numbers in the devices are generated independently and randomly. Choose $m \times n$ devices from the same batch. Form an $mn \times mn$ binary matrix M , with the i th column $M_i = (P^i)^T$. The matrix M is randomly generated.

Proof. Let r_j^i ($0 \leq i \leq m \times n - 1, 0 \leq j \leq m$) be n -bit binary numbers, $r_j^i = \pi_{j+1 \bmod m}(K_{D_{j+1 \bmod m}}^i)$ for $j \leq m-1$ and $r_m^i = K^i$ (where $K_{D_{j+1 \bmod m}}^i$ is the $j+1$ th random

number in the i th device and K^i is the key in the i th device. Assume $K_{D_{j+1 \bmod n}}^i$ and K^i are generated independently and randomly, so r_j^i ($0 \leq i \leq m \times n - 1, 0 \leq j \leq m$) are generated independently and randomly.

Let s_j^i ($0 \leq i \leq m \times n - 1, 0 \leq j \leq n - 1$) be $(m + 1)$ – bit binary numbers, the k th bit of s_j^i is determined from $s_{j,k}^i = r_{(j \times (m+1) + k - (j \times (m+1) + k \bmod n)) / n, j \times (m+1) + k \bmod n}^i$, i.e., s_j^i ($0 \leq i \leq m \times n - 1, 0 \leq j < n$) are the permuted result from r_j^i ($0 \leq i \leq m \times n - 1, 0 \leq j \leq m$). Clearly, the elements $s_{j,k}^i$ are generated independently and randomly.

Form an $(m + 1)n \times mn$ matrix S with $S_{i,j} = s_{(i - i \bmod m + 1) / m + 1, i \bmod m + 1}^j$. Since every element of S is generated independently and randomly, the probability that S^T with rank mn is about

Form an $mn \times (m + 1)n$ matrix T and an $mn \times mn$ matrix P :

$$T = \begin{bmatrix} T_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & T_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & T_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & T_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & T_1 \end{bmatrix} \quad P = \begin{bmatrix} P_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & P_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & P_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & P_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & P_1 \end{bmatrix}$$

where T_1 is an $m \times (m + 1)$ binary matrix in the form of (A.1) and P_1 is an $m \times m$ matrix in the form of (A.2). From Lemma 2 and 3, it is easy to see that both P and T are with rank mn .

Define an $mn \times mn$ matrix M' as $M' = P \cdot T \cdot S$. From Lemma 1, we know that M' is randomly generated since P and T are with rank mn and S is randomly generated.

The matrix M formed in Scheme 3 could be considered as being formed by exchanging the columns of M' . So the matrix M is randomly generated.

An Implementation of Bitsliced DES on the Pentium MMXTM Processor

Lauren May, Lyta Penna, and Andrew Clark

Information Security Research Centre,
Queensland University of Technology,
GPO Box 2434, Brisbane, 4000, Australia.
{l.j.may, l.penna, a.clark}@qut.edu.au

Abstract. In this paper we describe an implementation of the DES (and Triple-DES) algorithm using the bitslicing technique on an Intel Pentium MMX processor. Implementation specifics are discussed and comparisons made with an optimised C-coded DES implementation and an assembly language DES implementation. This paper sets the scene for future research of the inter-relation between design and implementation of the newer 128-bit symmetric block ciphers.

1 Introduction

Symmetric block cipher design is, by necessity, influenced by the technology to be used in the implementation of the cipher. The primary goal of a cipher is to provide security, with its optimisation of implementation being a very important secondary consideration.

Early cryptographic processing was generally a bottleneck in communications and its implementation in hardware was necessary to optimise its speed. Today commodity microprocessors are available which can, in general, process software implementations of ciphers quickly enough for most purposes. Of course there are benefits of hardware devices for performing cryptographic processing such as the secure storage of keys due to tamper-resistance. However, there are numerous applications such as web browsers and email encryption utilities where ciphers are implemented in software and hence, here we focus on optimisation issues in software implementations of block ciphers.

Many optimisation techniques tend to be a trade-off between storage and speed. The storage of large amounts of information may be acceptable on platforms that are unrestricted by memory limitations but may not be possible on other platforms where the amount of available memory is minimal (such as a smart card).

Optimised cryptographic code is generally either purchased from a vendor or written in-house. People who optimise cryptographic algorithms, complemented by an appreciation of architectures and platform instruction sets, possess a very specialised skill. As technological advances and platform diversity proliferate the demand for these skills increases accordingly. Optimised code and methods for

achieving it are generally closely-guarded secrets, with information in the public domain being scarce. In this and consequent papers, the authors' contribution is to make our findings and applications available in the public literature.

The Data Encryption Standard (DES) [3] has been the symmetric block cipher standard since 1976 and is the most commonly studied cipher due to its widespread use. Many different techniques can be used to improve the efficiency of DES (for example [2]). One recent technique which indicates a path to faster implementation of some encryption algorithms is the use of bitslicing. This technique was applied by Eli Biham in 1997 [1]. Section 2 discusses some implementation and optimisation issues with the Pentium MMX platform. Section 3 compares the C, assembler and bitsliced MMX DES implementations and investigates three potential S-box algorithms. Section 4 discusses some quite restrictive limitations of MMX bitslicing regarding modes of operation and the platform itself, whilst Section 5 concludes the paper.

2 Issues in Implementing Bitslicing on a Pentium

Bitslicing enables the DES S-boxes to be represented in software by their logical gate circuits. Bitslicing, essentially, encrypts one bit at a time. If the implementation platform allows for parallel data processing then, in effect, a number of single-bits are encrypted simultaneously. Intel Pentium MMX technology is one such platform that uses a 64-bit register for parallel data processing. These processors are very relevant to cryptographic application as they are in such widespread use, particularly in commerce which is one of the heaviest users of cryptography.

In the case of a 64-bit processor, bitslicing enables sixty-four datablocks to be processed concurrently. This section is concerned with implementing the complete DES cipher using the bitslicing technique. The Pentium MMX bitsliced code in this paper was produced by using the Visual C++ compiler with in-line assembly.

In his paper Biham described a new optimised bitsliced DES implementation on a 300MHz Alpha 8400 consisting of thirty-two 64-bit integer registers. As was shown on such a computer, it is possible to gain a significant speed-up over standard implementations using the bitslicing technique as sixty-four datablocks are encrypted simultaneously. Biham's implementation ran three times as fast as the fastest implementation at that time (Eric Young's *libdes* [8] which was designed for 32-bit architectures).

Section 2.1 summarises the MMX instructions employed in the bitsliced implementation and also notes some subtleties in their use. Section 2.2 discusses some Pentium-specific optimisation techniques.

2.1 MMX Instructions

Intel's MMX architecture was designed specifically to enhance the performance of advanced media, multimedia and communication applications. It achieves this

through the introduction of new dedicated 64-bit integer registers, a new instruction set which allows for data parallelisation and a superscalar architecture which enables instruction-level parallelisation.

The MMX instruction set adds fifty-seven new instructions. The six instructions used in the MMX implementations discussed in this paper are:

EMMS enables the MMX registers to be aliased on the floating point unit
 MOVQ moves a 64-bit word between registers/memory and registers/memory
 PXOR binary EXCLUSIVE-OR
 PAND binary AND
 PANDN binary NOT followed by binary AND
 POR binary OR

The EMMS, MOVQ, PXOR and POR instructions are straightforward and give no surprises in their application. The use of the PAND and PANDN instructions, however, contain subtleties which we now address.

The PAND instruction performs a binary AND on sixty-four bits of data from an MMX register or memory to an MMX register. For example, the code *PAND MM0,MM1* performs a bitwise logical AND of the contents of MM0 and MM1 and stores the result in MM0.

The PANDN instruction firstly inverts (complements) the sixty-four bits in an MMX register and then ANDs the inverted MMX register contents with another MMX register or memory. For example, the code *PANDN MM0,MM1* initially inverts the contents of MM0, and then performs a bitwise logical AND of the result and the contents of MM1. The result is stored in MM0.

Using the PANDN instruction requires multiple additional instructions when compared to storing a NOT value using C code and then using the PAND assembly instruction. This is because the value to be inverted must be in the first operand and this value is destroyed during the execution of the instruction, resulting in multiple additional instructions being required to retain the initial value in the first operand.

Bitslicing DES S-boxes requires the use of a bitwise logical AND and a bitwise logical NOT instruction that performs a bitwise logical NOT on a 64-bit data-block. As there is no MMX NOT instruction, it is necessary to use the PANDN instruction. This limits the processing speed of the S-boxes severely due to the multiple extra instructions required to maintain the state of the registers.

2.2 Pentium

The management of Intel's branch prediction technique is an important tool to be used in any attempt at optimisation on the Pentium. Branch prediction occurs when a probabilistic decision, based on past events, is made by the compiler as to the address of the next instruction. Optimising branch prediction techniques is most beneficial where the code uses *if/then/else* statements, particularly in connection with looping structures. In this paper's MMX implementation,

the use of *if/then/else* statements is kept to a minimum, so this optimisation technique is not beneficial to our implementation.

The dual-pipelining architecture in the Pentium enables two individual instructions to execute in two processors (the *u pipe* and the *v pipe*) simultaneously, referred to as pairing the instructions. Maximal pairing can result in significant reductions in the number of clock cycles needed to implement code sections. The *u pipe* is the master processor whilst the *v pipe* is the slave processor. If there is no pairing of instructions, all instructions are fed through the *u pipe* whilst the *v pipe* remains empty. If there is maximal pairing of the instructions, half the instructions are fed through the *u pipe* and the other half are fed through the *v pipe* simultaneously. Therefore the aim of optimising by pairing instructions is to write code in such a way that the compiler keeps both pipes full. This dual-pipeline instruction processing is called superscalar parallelism. There are certain restrictions on the type of instructions that can be paired, mostly to do with the complexity of the individual instructions and the limitations of the supporting hardware. This approach applied to each of the eight S-box implementations resulted in a reduction of, on average, 23.6% of the original number of clock cycles.

3 Comparison of DES Implementations

This section compares the C, assembly (Eric Young's C and assembly implementations [8]) and MMX implementations with respect to various techniques. Code optimisation is, in general, a function of both the algorithm and the implementation language conjointly. Several different techniques are presented and discussed in this section.

Section 3.1 discusses the initial and final permutations. As the implementation of the DES S-boxes is the overriding factor in its optimisation, several S-box Boolean function algorithms are investigated with regard to their suitability for MMX implementation. Section 3.2 discusses the applicability of MMX instructions to these S-box algorithms. Section 3.3 gives a single-round comparison. A complete DES implementation comparison is given in Section 3.4, while Section 3.5 addresses Triple-DES implementations.

3.1 Permutations

Permutations are inefficient in software. Patterns in the permutations make the use of look-up tables and streamlined operations possible. The SWAPMOVE technique, which we now describe, requires no memory and is extremely efficient for the IP, FP and PC1 permutations of DES. This technique is utilised in versions of DES available from the Internet (for example Eric Young's *libdes* [8]). Consider the following process:

```

SWAPMOVE(A, B, N, M)
T = ((A >> N)  $\oplus$  B) & M;
B = B  $\oplus$  T;
A = A  $\oplus$  (T << N);

```

In this process the bits in B, masked by M, are swapped with the bits in A, masked by $(M \ll N)$.

The IP can be performed using five SWAPMOVE operations, totalling thirty logical operations. This is an extremely efficient implementation technique. The permutation FP is performed by reversing the order of the five SWAPMOVE operations used in IP. PC1 can be performed in a similar manner, requiring sixty-four logical operations. The optimised C and assembler codes use the SWAPMOVE procedure, but the bitsliced implementation does not.

SWAPMOVE is a function based on the positions of the bits within a block. Although it is an extremely efficient technique for a C-coded implementation it cannot be used in the bitsliced implementation as the block bit positions are changed; therefore a standard IP permutation was used in the MMX code and, as can be seen in Table 1, it was faster than the optimised DES version that used the SWAPMOVE procedure. This speed-up is gained due to the 64-bit parallel processing bitslicing allows.

Table 1. Initial Permutation (IP)

Code	Clock cycles/ 64 DES data blocks
C DES	2659
Assembler DES	1541
Bitsliced DES	148

3.2 S-box Implementation

S-boxes that are implemented using Boolean functions are generally slower than when implemented using look-up tables however, when using them to process sixty-four word blocks in parallel, they give a more optimised S-box implementation.

There are multiple Boolean functions which can represent the DES S-boxes and be implemented efficiently. Schaumuller-Bichl’s *Method of Formal Coding* [6], Shimoyama’s algorithm [7] and Kwan’s algorithm [4] were implemented for comparative purposes.

Kwan’s algorithm is clearly the fastest when written in the C language (Table 2). All algorithms were also programmed using MMX instructions (Table 3). Kwan’s algorithm remained the fastest, although there was also a substantial speed improvement for Schaumuller-Bichl’s algorithm. Shimoyama’s algorithm written in MMX assembly language (Table 3) did not show significant speed improvement due to the multiple variables involved in the algorithm.

A note on timings: With each new Intel Pentium processor some instructions execute in a reduced number of clock cycles. In our experience we found that the difference between the Pentium II and Pentium III was not substantial. The timings for all tables in this paper were performed on a 500 MHz Pentium III. We found the Pentium with MMX processor to be, on average, twenty percent slower than the Pentium II/III.

Table 2. Comparison table of C-coded S-box algorithms

	Code	Clock cycles/ 64 S-box 1 inputs
1.	Schaumuller-Bichl (SB)	243
2.	Shimoyama (S)	235
3.	Kwan (K)	150

Table 3. Comparison table of MMX-coded S-box 1 implementations

	Code	Clock cycles/ 64 S-box 1 inputs
1.	Schaumuller-Bichl	137
2.	Shimoyama	226
3.	Kwan	90

Table 3 clearly indicates that the most suitable algorithms for MMX implementation are Schaumuller-Bichl's *Method of Formal Coding* and Kwan's algorithm.

Using Schaumuller-Bichl's algorithm, S-box 1 was implemented in four different ways (Table 4).

Table 4. Comparison table of S-box 1 implementations using the Schaumuller-Bichl algorithm

	Code	Clock cycles/ 64 S-box 1 inputs
1.	Using variables and PANDN	166
2.	Using minimal variables and PANDN	245
3.	Using variables and store NOT values	174
4.	Using minimal variables and store NOT values	137

In Table 4 *Implementations 1* and *2* are compared as they differ only in the number of variables used, as opposed to the number of registers used. This comparison indicates that using variables to store data, as opposed to trying to store them in the registers, makes for faster implementation as well as easier coding. Alternatively a comparison of *Implementations 3* and *4* (which differ only in that the latter uses fewer variables) indicates that using fewer variables is faster. Therefore, in general, the effect of the number of variables used is inconclusive. In Table 4, *Method 4* was the fastest, therefore it was used for all eight S-boxes in the final DES bitsliced implementation using Schaumuller-Bichl's algorithms.

Kwan's S-box 1 implementation was faster than Schaumuller-Bichl's implementation (Table 3), therefore Kwan's algorithms for the eight S-boxes were also implemented for the final DES bitsliced comparisons.

Table 5 gives the timings for each of the eight S-boxes for both algorithms. For each of the S-boxes Kwan’s algorithm produced the fastest MMX code. However, as will be discussed in Section 3.4, the full DES implementation using Kwan’s S-box algorithms actually executed slower due to pre-processing and overheads in execution.

Table 5. Eight individual MMX S-box implementations

S-box (64 inputs)	1	2	3	4	5	6	7	8
S clock cycles	137	111	123	172	152	124	140	135
K clock cycles	90	95	87	80	101	98	96	88

3.3 A Single Round of DES

A single round of DES for each of the four approaches was applied and the results compared. Table 6 indicates that the bitsliced implementation using the Schaumuller-Bichl S-box algorithms is the most optimal.

Table 6. One round

Code	Clock cycles/ 64 DES data blocks
C DES	17786
Assembler DES	14255
SB Bitsliced DES	10597
K Bitsliced DES	18831

3.4 Complete DES Implementations

The complete DES implementation includes all the previously-mentioned components. Table 7 indicates that the complete bitsliced DES using Schaumuller-Bichl’s S-box algorithms is 68% faster than the complete C-coded DES and 35% faster than the assembler DES. Timings were performed on a 500 MHz Pentium III.

Table 7. Complete DES implementation

Code	Clock cycles/MByte	Mbps
C DES	72857172	54.90
Assembler DES	58389896	68.50
SB Bitsliced DES	43405661	92.15
K Bitsliced DES	77130624	51.86

It is of interest that, even though the implementations of each of the individual S-boxes is, on average, 33% faster using Kwan's algorithms (Table 5), the final DES bitsliced implementation (Table 7) using Kwan's S-box algorithms was 45% slower than using Schaumuller-Bichl's S-box algorithms. This significant difference is attributed to the fact that Kwan's S-box algorithms require substantially more variables to implement (and hence more pre-processing and overheads) than Schaumuller-Bichl's S-box algorithms.

This finding indicates that, using MMX technology, it is important to balance the number of variables and the number of logic gates used in the algorithm due to the limited number of MMX registers and the limited instruction set. If the MMX platform had thirty-two MMX integer registers instead of the currently-available eight, the full DES implementation using Kwan's S-box algorithms should be faster than the Schaumuller-Bichl algorithms.

3.5 Triple-DES Implementation

Triple-DES was implemented using the procedures from both bitsliced DES implementations. From a programming perspective, the only difference between Triple-DES and standard DES (other than tripling the number of DES procedures performed) is that Triple-DES does not require the IP and FP permutations between the first and second single DES, and the second and third single DES, since they are the inverse of each other. The initial permutation before the first single DES and the final permutation after the third single DES are still required.

The Schaumuller-Bichl bitsliced Triple-DES (Table 8) is the fastest implementation. Timings were performed on a 500 MHz Pentium III.

Table 8. Triple DES implementation

Code	Clock cycles/MByte	Mbps
C DES	183315583	21.80
Assembler DES	140440785	28.48
SB Bitsliced DES	127064841	31.48
K Bitsliced DES	185454810	21.00

4 Limitations

There are some limitations in using bitslicing both in general and also on a Pentium MMX platform. The bitslicing technique has some restrictions in general in respect to common modes of block cipher implementation, which are discussed in Section 4.1. The Pentium MMX platform has some restrictive limitations for the bitslicing application when compared to the Alpha 8400 (Section 4.2), but it is in much more widespread use than the Alpha and so is a more acceptable block cipher implementation platform. Section 4.3 explains the requirements of data-format conversion for the bitslicing technique.

4.1 Modes of Operation

Bitslicing implementations are suited to encryption and decryption using *Electronic Code Book (ECB)* mode. Since sixty-four data blocks are encrypted or decrypted simultaneously using the bitslicing technique, the method is not suitable for use with traditional block cipher modes where the current data block is reliant on the output from a previous data block encryption or decryption. These modes are *Cipher FeedBack (CFB)* encryption, *Cipher Block Chaining (CBC)* encryption, and *Output FeedBack (OFB)* encryption and decryption. With both the *CFB* and *CBC* modes the ciphertext is available (without the necessity of having to reproduce it in the decryption process), so the bitslicing implementation can be used for decryption in these modes. Traditional feedback and chaining modes of operation are used in most applications of DES, so this is a major limitation of the bitslicing implementation technique.

In his paper Biham proposed a CBC-like mode where the IV is 4096 bits and the entire 4096-bit ciphertext (64×64 -bit blocks) block is XORed with the ensuing 4096-bit plaintext block. This mode would overcome some of the shortcomings associated with the *ECB* mode of operation.

There would, however, be security problems associated with using this CBC-like mode operated on a 4096-bit block. For example, the traditional CBC mode operated on the standard 64-bit block offers a means of producing a Message Authentication Code (MAC). One of the intrinsic security features of the MAC (produced by DES encryption in traditional 64-bit CBC mode) is that the MAC is a function of every bit of the message or, conversely, every bit of the message used to generate the MAC is diffused across the entire MAC. In the case of a 4096-bit CBC-like mode this would not be the case since diffusion of a particular bit of the message would be restricted to a particular 64-bit subblock and not the entire 4096-bit block, as is a basic requirement of a MAC.

ECB mode however is still used in some applications, and these applications will directly benefit from the DES bitslicing implementation.

4.2 Pentium MMX Platform

A limiting restriction of the Pentium is that only eight MMX registers are available. A DES bitsliced implementation requires a minimum of six registers, one for each S-box input bit x_j . The remaining registers are used as accumulators and for other intermediary tasks. In the case of the Pentium this leaves only two registers for these housekeeping tasks, which is insufficient for producing optimised code.

Where the available MMX registers are inadequate for the necessary tasks the programmer must use non-MMX 32-bit registers. Moving data to and from the non-MMX registers requires more use of the MMX move command which increases the number of instructions necessary in the implementation, as well as the extra overheads involved with swapping between a 64-bit and a 32-bit word size. The Alpha 8400, having four times as many integer registers as the Pentium MMX, would not have these restrictions.

Another restriction on programmers is that the MMX instructions accept only two operands. This necessitates more frequent moving of data within the registers and, consequently, produces code with more instructions. The restrictive PANDN instruction and the absence of a 64-bit logical NOT instruction (Section 2.1) are also limiting factors.

Biham's fastest DES bitsliced implementation was 137 Mbps on a 300MHz Alpha 8400 processor. The fastest 500MHz Pentium MMX bitsliced implementation in this paper is 92 Mbps, which is a respectable result given the limitations of the platform and its instruction set, especially the minimal number of MMX registers.

4.3 Data Conversion

The data for the bitslicing implementation needs to be read into the registers in a non-traditional format. The inverse of this initial reformatting must also be performed at the end of the data processing. There is an overhead cost involved with this reformatting. We did not include the cost of data preparation or reformatting in any implementation. When used in situ, the code would be executed by a call from the Application Program Interface (API) which would incorporate any necessary reformatting before and after encryption and decryption.

5 Conclusion

In this paper the technique of bitslicing is applied to the DES algorithm on the very widely-used Intel Pentium MMX processor at a speed of 92 Mbps. Pentium implementation and optimisation issues are discussed. Some application limitations of bitsliced DES are identified, and a comparison of the C, assembler and bitslicing approaches is made.

Our research goal is concerned with implementation issues of symmetric block ciphers and how they affect the design of ciphers.

In this paper we have discussed a wide range of implementation issues of the DES symmetric block cipher. Our future research will concentrate on investigating implementation issues with respect to the new 128-bit block ciphers (for example, the new AES candidates [5]), and how these implementation issues affect the design decisions of the ciphers themselves. This work will build on the framework set out in this paper.

References

1. E.Biham, *A Fast New DES Implementation in Software*, 4th International Workshop, FSE'97, Israel, January 1997; Proceedings, Lecture Notes in Computer Science, Springer, Vol 1267, pp 260-271.
2. M.Davio, Y.Desmedt, J.Goubert, F.Hoornaert and J.J.Quisquater, *Efficient Hardware and Software Implementations for the DES*, Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196, Springer-Verlag, pp 144-146.

3. Federal Information Processing Standards Publications, FIPS PUB 46-1, *Data Encryption Standard*, USA.
4. M.Kwan, *Bitslice DES, S-box Implementation*, March 1998, <http://www.darkside.com.au/bitslice/sboxes.c> .
5. National Institute of Standards and Technology (NIST), *Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES)*, Federal Register, Volume 62, Number 177.
6. I.Schaumuller-Bichl, *Cryptanalysis of the Data Encryption Standard by the Method of Formal Coding*, Advances in Cryptology, EUROCRYPT'82 Proceedings, Springer, Verlag, 1992, pp 235-255.
7. T.Shimoyama, S.Amada, S.Moriari, *Improved Fast Software Implementation of Block Ciphers*, 1st International Conference, ICIC'97, China, November 1997, Proceedings: Lecture Notes in Computer Science, Vol 1334, pp 269-273.
8. E.Younglibdes, <http://www.SSLeay.org>.

Securing Large E-Commerce Networks

Panagiotis Sklavos¹, Aggelos Varvitsiotis², and Despina Polemi²

²Institute of Communication and Computer Systems

¹Department of Electrical and Computer Engineering

National Technical University of Athens

Hroon Politechniou 9

Zografou, Athens, Greece 157 73

Tel: 30-1-772 2466

Fax: 30-1-772-3557

psklavos@softlab.ntua.gr, avarvit@cc.ece.ntua.gr,
polemi@softlab.ntua.gr

Abstract. The Chambers of Commerce (CoCs) have to integrate into the emerging Public Key Infrastructure (PKI) which will be a prerequisite for secure Electronic Commerce (E-commerce). To fulfil the future requirements of E-commerce the CoCs need a secure communication channel between each other and the ability to ensure electronic documents of customers to be verified by any third party. In this paper we present a new Trusted Third Party (TTP) service which allows a large chain of different CoCs, that act as Certification Authorities (CAs), to interoperate.

1 Introduction

The European Commission under the Fourth Framework Programme in the Telematics Applications Programme for Administrators has funded a project entitled COSACC (Co-ordination of Security Activities between the Chambers of Commerce)¹. COSACC which is a two-year project, started on July 1998. It aims to identify current and future business scenarios for Chambers of Commerce (CoCs) which can be handled electronically, permit the CoCs to act as a vehicle for international electronic commerce, and provide a secure link between CoCs in order to enable them to take their primary business into an electronic realisation.

This paper presents results from this ongoing project, in particular a new Trusted Third Party (TTP) service called the Trust Transitivity (TT) service, developed by the authors, in order to address the need for interoperation between different Certification Authorities (CAs).

¹ The third author is the Technical Manager of the COSACC project.

Traditionally the CoCs are Trusted Third Parties. Today their main operational tasks are:

- Running the business register (official company registers).
- Issuing certificates of origin (certifying the origin of products).
- Issuing carnet ATA (customs guarantees for temporarily exported goods).

However, the organizational structure of the European CoCs is different from country to country. In some countries they are public law organizations (e.g. Greece, Netherlands) in other countries they are private law organizations (e.g. Denmark, Switzerland) which act on behalf of the state for specific tasks. In Switzerland the business register is not run by the CoC.

For specific purposes, the CoCs are organized on an international level. For instance with respect to the carnet ATA they run an international organization according to the carnet ATA convention managed by the International Bureau of Chambers of Commerce (IBCC).

E-Commerce [5,6,9] is now a new challenge for the CoCs. On the one hand, the traditional business based on paper documents is diminishing and must be supported by electronic documents and communication. On the other hand, the replacement of paper documents by electronic files offers new opportunities for the CoCs to act as Trusted Third Parties [8]. The necessary technical infrastructure [7, 29] and the tools to cope with this challenge exist or can be implemented:

- The digital signature which guarantees authenticity of the sender, integrity of the document and non-repudiation of origin. Confidentiality can also be provided with the same cryptographic algorithms.
- A Public Key Infrastructure [33] as a prerequisite for the proper organizational functioning and the legal acceptance of the digital signature.

The commercial purpose of the COSACC project is to enhance of the existing services and to adapt them to E-commerce. COSACC deals with

- the implementation of a secure electronic channel between CoCs over existing public networks (e.g. Internet) for the exchange of documents and
- the electronically sealing of entities (documents, programs, web sites) by a specific CoC, in a way that the seal can be verified by any third party using World Wide Web (WWW) techniques.

In this paper the various identified security threats, requirements and security services for the secure interconnection of the CoCs, which will act as TTPs, are described. In particular, this paper has been organized as follows: Section 2 provides an overview of common threats in E-Commerce networks and of the security considerations in the CoCs. Section 3 discusses the Core TTP services that provide an answer to the security requirements of the CoCs. Section 4 presents the functionality and the implementation of the TT service. Finally, Section 5 draws conclusions and directions for further research.

2 Threats and Security Considerations in the CoCs

The lack of a technological infrastructure that will ensure secure transactions hinders the interoperation between European CoCs and their evolution as core components of a pan-European E-Commerce [1] infrastructure.

Traditional business practice relies mainly on the trust between the involved parties. As a result any suggested technological solution for the E-Commerce should provide the same or higher level of trust to the (digitally) involved parties. In order to provide this level of trust a network of TTPs [11] must be evolved. European CoCs are the dominant candidates for undertaking this role [14]. The reasons why European CoCs are the dominant TTP candidates for the evolved pan-European E-Commerce Information infrastructure include:

- Enterprises have already developed strong trusted relations with CoCs in each country with regards to traditional commerce. As a result CoCs possess a large enterprise related information base in each country. A pan-European CoC network would facilitate the formulation of the pan-European Electronic Commerce Information infrastructure. In addition CoCs are also linked together around the world through the International Chamber of Commerce (ICC).
- The information, which is provided through the Internet for Electronic Commerce purposes, must be accurate and reliable [13]. CoCs, and especially ICC, must undertake a role contributing to the harmonization and availability of global terms and conditions and the provision of global principles of interactive advertising ethics.
- Electronic Commerce greatly involves legal terms and guidelines in case of disputes. Trusted and neutral adjudication bodies [15] should deal disputes. Having gained enterprise confidence, the CoCs are best candidates for this role.
- CoCs have been granted authority by their respective governments to act as the official national business registers. A secure pan-European link between CoCs will provide a secure global business register infrastructure. The expansion of Electronic Commerce will be greatly influenced by the level of trust the enterprises themselves as well as the consumers will put on this infrastructure.
- Finally, enterprises are more and more dependent on the automation of their transactions and exchanges due to transaction cost reduction. In addition, the progress in telecommunications offers great potentials to enterprises, which pursue involvement in international trade and marketing. As a result, the automation of enterprise-CoCs transactions must be automated, reliable and cost beneficial not only on the national level but also on the international one.

The CoCs can provide the infrastructure in order to secure the operation of newly emerging applications such as E-Commerce [28]. Common threats [3,10,19,32] these application face include:

- **Monitoring of communication lines:** networks are susceptible to any station monitoring all the traffic. Encryption techniques constitute the main response.
- **Traffic analysis:** logging source and destination addresses, an individual can built up patterns of traffic. Again the classical way of preventing this is with encryption.
- **Non-authorised modification of information:** injection of new and modification of trespassing data can be confronted with message digests.
- **Denial of information received or sent :** a sender can ignore the transmission of a certain message and a recipient can ignore an unwanted message and deny its arrival. Non-repudiation processes face this kind of threat.
- **Masquerade of identity :** digital signatures and encryption techniques can face this threat.
- **Denial of Service (DoS) and Misuse of resources:** legitimate access to a network or information system can be denied and system degradation occurs.

The setup of access control and monitoring of system resources can handle such cases.

- **Key Personnel and physical insecurity:** ensuring that proper procedures are adhered to by all key personnel can help to minimize such threats.
- **Scalability:** Large PKIs face scalability problems arising from the management of numerous certificates, the interoperation and verification need between different CAs.

These threats can be confronted with success by a security infrastructure implemented and maintained by CoCs. A security solution based on the establishment of Trusted Third Party services provides an answer to the security considerations of CoCs.

3 Core Trusted Third Party Services (TTPs)

A TTP has been defined by ISO/IEC [22] as a security authority or its agent trusted by users with respect to security-related activities, e.g. to support the use of digital signatures and confidentiality services.

The architectural design that was selected [4,16,34] to implement the required TTP infrastructure in the case of COSACC is a non-hierarchical structure (Fig. 1).

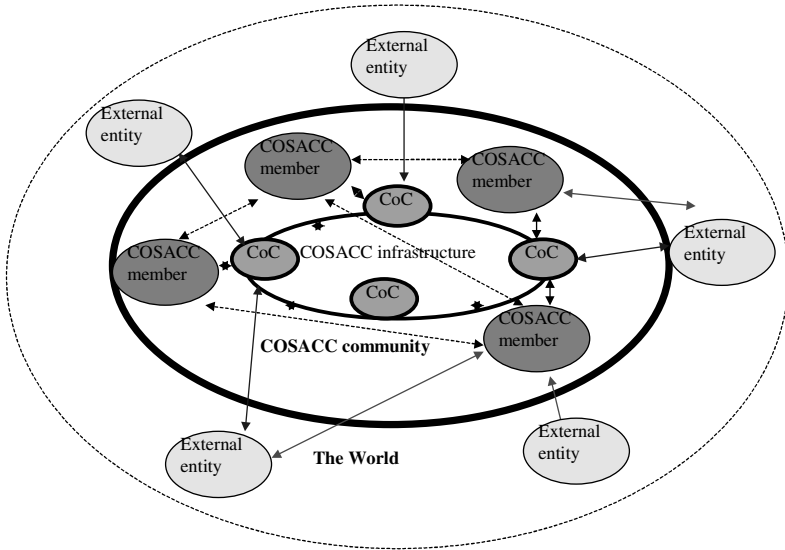


Fig. 1. COSACC Architectural design

The advantages of such a structure with respect to COSACC requirements can be summarised to the following features :

- Openness in relation to various user needs and policies in different domains,
- easier establishment,
- flexibility on legal requirements,
- easy transformation to a hierarchical structure (if needed).

The TTP services that constitute the COSACC Security Solution have been divided into two large sections: the *Core Services* and the *Value Added Services*.

Based on the type of information transactions between CoCs, the *Core Services* that must be supported by a TTP targeted to provide a security platform for CoC interconnection, include:

- **Registration** for the initiation of secure commercial transactions.
- **Digital signatures** [18, 25] for the authenticity and the integrity of sensitive contractual documents and for increasing the level of trust in the transactions with its customers.
- **Encryption** [2,17,22,26,30,31] for confidential financial reports, inter-department communication and technical information.
- **Key and Certificate management service** [21] is required for controlling the keys and certificates of the employees, as well as it is required by the company's directors in order to have access to all the documents of the company that they are signed and/or encrypted by their employees.

- **Directory Services** [12] that constitute a primary tool for storing and distributing certificates, CRLs and other information used in a trust infrastructure, such as personal details, functional components etc. The directory service, known also as the ‘public key directory’ (PKD) is an important part of key and certificate management within the CoC TTPs.
- **Archiving service** is also necessary for facing disputes arising between other companies or customers.
- **Authorization service** for transferring and granting of rights from a high level department to a lower level department or employee, enabling her/him to perform a specific transaction.
- **Audit** for monitoring the behavior of the system in order to assure that it operates according to certain predefined conditions.

Whereas the *Core Services* provide basic security to the CoC network, the *Value Added Services* aim to enhance the functionality of the TTPs and confront the threats that arise with E-Commerce transactions. The *Value Added Services* consist of: the *Time Stamping* service, the *Digital Seal* service and the *Trust Transitivity* (TT) service.

Time Stamping [1] uses the attachment of data and time on a document in order to prove that it has been issued on a particular date and time. The aim of the service is to prove the transmission the uniqueness of the document. Along with digital signatures, time stamps can be used to build a reliable non-repudiation service needed in several instances of the CoC operations dealing with payment orders, commercial invoices, custom or VAT declarations.

The **Digital Seal** service is the measure for authenticity and certification of Web pages. A COSACC Digital Seal is provided to all certified members.

The **Trust Transitivity** service provides the means for interoperation between TTPs and verification of their relationships. The TT service is described in the following section as the authors have developed it.

4 Trust Transitivity Service

The deployment of a large-scale network of CoCs leads to the establishment of a PKI with many TTPs. In such infrastructures, interoperation and transparency are important. These needs are addressed by the Trust Transitivity service (TT).

The TT service is responsible for providing the interoperation between different trust domains. A *trust domain* is formed by all certificates issued by the same root. We define *trust community* as a set of disjoint trust domains. The TT service operates over the World Wide Web (WWW). The purpose of the TT service is to form trust communities by creating chains of trust. The trust chains are constructed by the cross-certification between different trust domains. The Trust Transitivity service provides also a mechanism for the verification of the cross-certificate chains. End-users consult the service in order to accept formerly untrusted CA certificates in a secure manner.

Within the COSACC trust community, the so-called TT Top Level Certification Authority (TLCA) handles cross-certification. The TLCA is acting as a mediator in every cross-certificate path (Fig.2).

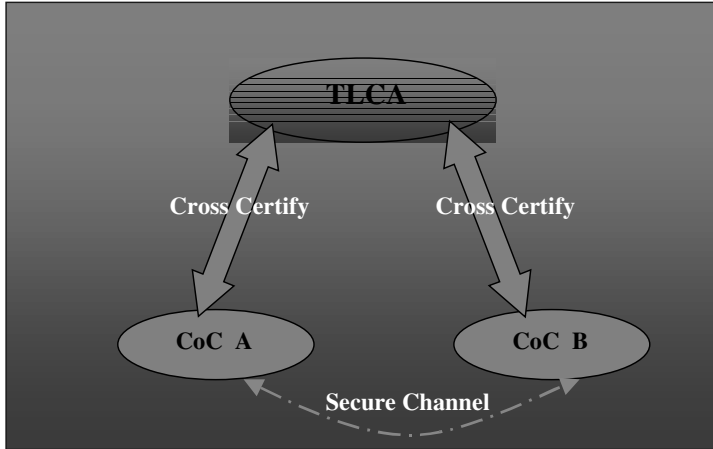


Fig. 2. Cross-Certification Architecture

By having the TLCA cross-certify with each COSACC TTP, we create a common reference point for all trust chains. This reference point is not only used for the construction of the cross-certification chains, but also for the verification of these cross-certification chains. In fact, the TLCA is also responsible for the verification of every cross-certification chain within COSACC. It handles the revocation of the COSACC Certification Authorities through self-maintained Authority Revocation Lists (ARLs) and provides the most up-to-date information concerning the end points of a cross-certification path (i.e. the CAs).

The Trust Transitivity service is implemented in the COSACC security solution in three distinct phases: Registration, Cross-Certificate Generation and Cross-Certificate Path Verification.

The first phase, Registration, has been implemented through e-mail. Registration is mandatory for a TTP in order to cross-certify with TLCA and become a valid member of the trust community of COSACC. The Administrator of the CA wishing to register, sends an e-mail to the Administrator of the TLCA providing the Distinguished Name of the CA, the e-mail address of the Administrator of the respective CA and a passphrase used for later authentication. The passphrase can be sent in plain text or encrypted. This is because Registration offers the flexibility to the CA Administrators to change the original passphrase and thus maintain confidentiality. In case the CA does not operate a Directory Server, the CA Administrator must include in his e-mail a URL in the CA's Web Server from where the root self signed certificate of the CA can be downloaded at runtime. The selection of self signed certificates for the CAs of the CoCs arise from issues such as: strengthening the trust among CoCs, for flexibility and time-saving reasons and finally from the requirements of the Verification phase as imposed by the PKI software used [37].

The Registration service also includes a service that allows the CA Administrators to change the passphrase. This is achieved by a Web interface running exclusively over HTTP-over-SSL [35] which prompts the CA Administrators to change their passphrase. In order to validate the change of passphrase, the service first authenticates the CA Administrator by using the CA Administrator's old credentials and the DN of the CA and by verifying an existent entry in the TLCA's database.

The second phase and most important phase of the Trust Transitivity service, is the Cross-Certificate Generation service. It is the part that generates trust relationships that lead to the creation of a trust community. Trust relationships are realised by cross-certificates.

A cross-certificate is a data structure that consists of a pair of X.509 certificates [20, 23], named the *forward* and the *reverse* component. The particular property of the two certificates bound in a cross-certificate pair is that the *Issuer* field in the forward component is exactly the same as the *Subject* field in the reverse component and vice versa. Both the Issuer and the Subject must therefore be Certification Authorities, which certify one another.

The Cross-Certificate Generation is a service offered only to registered CA Administrators through a Web interface running both over plain HTTP [2] and HTTP-over-SSL [35]. The service consists of three basic steps:

1. The first step involves the issuance of a Certificate Signing Request (CSR) [24] by the TLCA, after the CA administrator has authenticated by providing the appropriate credentials.
2. The second step involves the signing of the CSR from the CA that wishes to cross certify with TLCA. The CSR is signed using the private key of the CA. Thus, the forward component of the cross-certificate is created.
3. The final step involves the generation of the reverse component of the cross-certificate by the TLCA. The CA Administrator posts the signed CSR to the service and provides authentication credentials before requesting a cross certificate. If all checks succeed, the service generates a cross-certificate for the CA.

Cross-Certificate Generation has been built upon the **Open Secure Certificate ARchitecture** (OSCAR) [37] of the Security Unit of the Distributed Systems Technology Centre (DSTC) and the Open Source toolkit that implements the Secure Sockets Layer (OpenSSL) and is a descendant of the SSLeay implementation of the Secure Sockets Layer (SSL) protocol from CryptoSoft. The TT Cross-Certificate Generation uses a set of CGI scripts in order to interface with OSCAR and OpenSSL. It utilises either the Web or the Directory for the retrieval of the CA's self-signed root certificate. The location of that certificate is known to the service through the Registration phase. The retrieval of the certificate is done at runtime. Runtime is essential, because if network is not available, the Cross-Certificate Generation does not proceed. After having retrieved the CA root certificate, the service uses OSCAR to decode the certificate, to verify the CA's signature in the forward component as explained in step 2 above, to create the reverse component and finally to generate the cross-certificate.

The third phase of the TT service is the Cross-Certificate Path Verification. Cross-Certificate Path Verification can be considered as a stand-alone service and it is not restricted to the members of a trust community, but rather, it is a publicly available service, offered to all members of the Internet community by the Web Server of the TT service. Its purpose is to verify that there exists a valid trust chain between two CA's.

A user who wants to verify the CAs trust relationship (Fig. 3) has to provide to the service input service information concerning the location of the root and cross-certificates (e.g. DN, URL). The user is assisted to do so by the interface of the service, which provides a list of CAs belonging in the same trust community. The procedure followed by the service to verify the cross-certification path is totally transparent for the end-user.

The mechanism for validating the trust chain depends on real time verification. The PKI tools of OSCAR and OpenSSL have been used to validate a cross-certificate chain. The procedure comprises two basic steps: the first step is the verification of the chain from a trusted CA up to the TLCA and the second step is the verification of the chain from the TLCA to the untrusted CA.

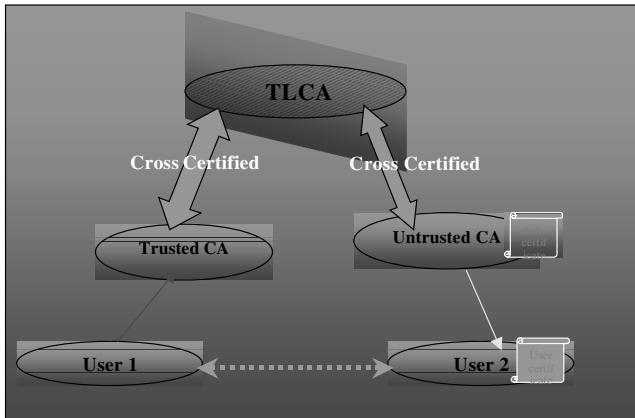


Fig. 3. Certificate Path Verification

Both steps are realised by the validation of the signatures in the self-signed certificates and cross-certificates of the entities involved. Signature consistency is not the only requirement for the verification of the chain. Validity Dates, proper Distinguished Names, Critical X.509 Extensions, etc. are also checked. All verifications take place after the certificates have been downloaded. The retrieval of certificates is done at runtime. The verification service denies to proceed if retrieval of certificates at runtime is not possible.

Revocation is offered for Certification Authorities only. An ARL (Authority Revocation List) is maintained by the Trust Transitivity TLCA and contains entries for revoked Certificate Authorities. The ARL does not need to be published since verification is done at runtime. In case a CA certificate is revoked, the verification fails warning the user and reporting specific information.

If a trust chain is verified successfully, the user is provided with information concerning the formerly untrusted Certification Authority. This information is extracted using the OpenSSL toolkit and includes the DN name of the CA, the Public Key and the fingerprint of the certificate. The service also prompts the user to install the newly verified certificate to a local certificate store. After successful verification, the certificate of the formerly untrusted CA is cached in a location in the secure part of the TLCA Web server and the user downloads the certificate over HTTP-over-SSL. Thus, the risk of a spoof attack against the real origin of the CA certificate (i.e. the CA itself) is minimized, since both the downloading and verification at real time ensures the user for the authenticity and validity of the certificate.

5 Conclusions-Further Research

In this paper we presented a new Trusted Third Party (TTP) service, developed in order to make different Certification Authorities (CAs) interoperable based on cross-certification and real time verification of cross-certificate chains. The verification of cross-certificate chains currently supports only Certification Authorities. It needs to be enhanced by an on line mechanism like the Online Certificate Status Protocol (OCSP) [27] in order to provide information concerning the status of the personal certificates of the end-users. Furthermore, the issue of interoperation in more complex hierarchies of CAs remains to be addressed. Integration with existing communication tools will enhance user friendliness and transparency. Although during the COSACC project the certificate policy issues were resolved by a contractual agreement between the members of the COSACC consortium, further study on policy issues needs to be conducted.

References

1. Adams C., Chain P., Pinkas D., Zuccherato R., 'Internet Public Key Infrastructure, Part V: Timestamp protocols', PKIX Working Group Draft, March 1998
2. ACTS/SEMPER Project Secure Electronic Market Place for Europe, 1995
3. ANSI X3.92 Data Encryption Standard
4. Ahuja, V. "Network & Internet Security "Academic Press, NY 1996.
5. Architecture for Public Key Infrastructure (APKI), Draft 1, The Open Group, May 1997.
6. Camp L.J, Sirbu M. „Critical Issues in Internet Commerce“ IEEE Communications Magazine, 58-62, 1997.
7. CEN/CENELEC/ETSI ITAEGV/N231, M-IT-06, Taxonomy and Directory of European Standardization Requirements for Information Systems Security, Issue 2.1, October 1994.
8. Colleran, A. „Standardization Issues for the European Trusted Services -ETS, Quercus Information, May 1997
9. Communications of the ACM Special Issue: Electronic Commerce, vol.39, no.6, June 1996
10. COMPUTER „Network Security“ Special Issue vol.31, no.9, 1998.

11. Froomkin A. „The Essential Role of Trusted Third Parties in Electronic Commerce, October 1996.
12. Goodman D., Robbins C., 'Understanding LDAP & X.500', Sep 1998.
13. Gre, P., Rotenberg, M. „Technology and Privacy: The New Landscape“ MIT Press 1998
14. INFOSEC/ETS II/BESTS Project Business Environment Study of Trusted Services 1998
15. INFOSEC/ETS'96: „Legal, Ethical & Regulatory Issues concerning the TTPs and Digital Signatures“.
16. INFOSEC/TTP Project, Trusted Third Party Services: Functional Model, Muller P. (Ed.), Ver. 1.1, Bull Ingenierie, France, December 1993.
17. ISO/IEC 10118-2:92 „Hash Function for digital signatures Part2: Hash Functions using a symmetric block-cipher algorithm“
18. ISO/IEC 9796:91 „Information Technology- Security Techniques-Digital Signature Scheme giving message recovery“
19. IEC DIS 11577 Network Layer Security Protocol
20. IEC DIS 9594-8 X.509 The Directory Authentication Framework
21. ISO/IEC JTC 1/SC27 N691, Guidelines on the use and management of Trusted Third Party Services, August 1993.
22. ISO/IEC JTC1/SC27/WG2 Security Techniques, Security Mechanisms DIS 9796
23. ISO/IEC X.509 RSA Public Key Certificates
24. Kaliski, B., „PKCS 10 Certification Request Syntax Standard“, RFC 2314 March 1998
25. Kaliski, B., „PKCS 7 Cryptographic Message Syntax Standard“, RFC 2314 March 1998
26. Menezes, van Oorschot, Vanstone „Handbook of Applied Cryptography“, CRC Press, 1996.
27. Meyers M., et. al „X.509 Internet Public Key Infrastructure On Line Certificate Status Protocol (OSCP)“, June 1999.
28. Pfleeger S.L., "A Framework for Security Requirements", Computers and Security, Vol. 10, No 6, pp. 515-523, 1991.
29. Polemi D, "Securing E-Commerce" Proceedings of the International e-Commerce Conference, IECC'99, 1999.
30. Schneier, B. "Applied Cryptography, Protocols, Algorithms and Source Code in C", J. Wiley and Sons Inc, Second Edition, 1996.
31. Simmons G.J. „Contemporary cryptography: The science of information integrity“ IEEE Press 1992
32. Stallings W. „Network and Internet Security: Principles and Practice“ Prentice Hall 1995
33. The Burton Group, 'Public Key Infrastructure', Network Strategie Report, 1997
34. Telematics Engineering/ICE-TEL Project, Architecture and General Specifications of the Public Key Infrastructure, COST, September 1996.
35. T. Dierks et al „The TLS Protocol v.1.0“, June 1999
36. R.Fielding et.al. „Hypertext Transfer Protocol---HTTP/1.1“, June 1999
37. Open Secure Certificate Architecture (OSCAR), Security Unit, Distributed Systems Technology Centre, Australia, <http://oscar.dstc.qut.edu.au>.

Passive Entities: A Strategy for Electronic Payment Design

Ernest Foo and Colin Boyd

Information Security Research Centre
School of Data Communications
Queensland University of Technology
Brisbane, Australia
{ernest, boyd}@fit.qut.edu.au

Abstract. The participating entities in an electronic payment usually include an overseeing bank as well as a customer and a merchant. Many electronic payment schemes are designed so that the bank is off-line during the actual payment between the customer and the merchant. This paper introduces the concept of passive entities which allows the protocol designer to allocate the level of processing required by an entity during a particular phase of the payment scheme. To illustrate the use of passive entities, a new electronic payment scheme is developed which has the customer assigned as the passive entity during payment. This new protocol is more efficient than similar protocols. A variation of this protocol is also described in which the identity of the customer is kept anonymous from the merchant.

1 Introduction

Most of the secure electronic payments conducted on the Internet today use the Secure Socket Layer (SSL) protocol [13] to ensure that messages are exchanged with integrity and in a confidential manner. SSL also provides server authentication. But the security provided by SSL is often not all that is required in a payment transaction. Electronic payment schemes which sit in the application layer are required to provide additional services such as non-repudiation, dispute resolution, fair exchange and anonymity.

To provide these services electronic payment schemes are in most part based on forms of physical payment. The particular architecture that a payment scheme is based on is called a transaction model. The transaction model defines the number of entities required by the payment scheme and how these entities interact with each other. The transaction model may also limit the characteristics and payment services which are available in the payment scheme.

The majority of electronic payment schemes [2,3,4,5,6,7,8,9,10,14,15,16] etc. are based on the traditional credit card transaction model. This model consists of a framework of three entities: the customer, the merchant and the bank. These parties usually conduct several exchanges to achieve secure electronic payment.

In this paper we present the new concept of passive entities which create interesting electronic payment systems that are not restricted to the traditional credit card transaction models. These new transaction models provide an opportunity to shift the burden of computation and storage and improve the efficiency of the payment transaction. They also allow flexibility for different application scenarios.

1.1 Related Work

The amount of work which concentrates specifically on the design of transaction models for electronic payment systems is not extensive. Many electronic payment schemes are designed in an “ad hoc” manner or using general secure protocol design principles such as the ones presented by Abadi and Needham [1]. There are several examples of different transaction models being employed by various electronic payment schemes.

The older and more traditional electronic payment schemes are based on a funds transfer or credit card transaction models. These transaction models allow for a payment request from the merchant to be authorized by the customer. The bank on verifying the validity of the authorization directly transfers funds from the customer’s account to the merchant’s account. The funds never leave the secure banking environment. The Secure Electronic Transaction (SET) [14,15,16] protocol and iKP [2] protocols are based on this transaction model.

A token based transaction model is required for off-line anonymous exchanges. This transaction model is based on the exchange of physical cash. The token allows funds to be anonymously withdrawn by customers, paid to merchants and deposited at banks. If a token is lost it usually means that the value is also lost. Most electronic cash schemes such as Brands’ Internet cash scheme [3,4] and Chaum’s eCash [7,18] scheme are based on this transaction model.

Foo and Boyd developed a voucher scheme [11] which uses a different transaction model. They identify that within a transaction model, all entities have to agree to commit to a transaction. This achieves the accountability or non-repudiation property required by most electronic payment. The commitment is usually represented by a bit string of data which is transmitted from entity to entity. A model of the commitment data was developed which enabled the design of new transaction models. The voucher payment scheme is the result of employing an alternate flow of this commitment data.

Kelsey and Schneier [17] also developed a purchase order payment scheme which uses a unique transaction model. The transaction model used in their scheme is “buyer driven”, thus it transfers most of the processing load to the buyer, allowing the customer to determine the conditions of the sale rather than the merchant which is the case for most of the payment schemes described previously.

1.2 Our Approach

Original electronic payment schemes can be developed by assigning one or more entities in the transaction model as being passive. Meaning that the entity does not conduct any cryptographic processing but may be required to transmit and receive messages to other entities. There is no limitation as to which parties may be passive as either the customer, merchant or bank may be passive during the payment phase. We illustrate the usefulness of this new design concept by presenting an efficient payment scheme which has a passive customer.

This new protocol has many uses and can be equated to the real world equivalent of a classified index or stock exchange. Our protocol allows customers to advertise an order for a particular item or service. In the case of a stock exchange the customer would identify a particular company's shares she would like to purchase and at what price she is willing to buy them. The customer could also specify how long she is willing to make this offer available. The first step of the protocol requires the customer to register with the bank and then create a specially formatted purchase order. After establishing and posting her order the customer is not required to do any further on-line processing. The customer can publish her order on a web page or FTP site. Merchants, in this case stock brokers, are then able to browse these orders at leisure. This situation is ideal for merchants to employ software agents to seek suitable customers for their products. Once a merchant (or his software agent) decides to fill an order he encrypts the electronic goods, in our example the appropriate share certificate, and sends it to the bank. The bank then passes the filled offer back to the customer. Our protocol has the advantage of allowing the bank to settle disputes.

We regard the following as the main contributions of the current paper.

- The identification of a new property, namely passive entities, within the existing electronic payment scheme framework. This property can be employed to design new payment schemes.
- A new electronic payment scheme which assigns the customer as a passive entity. Thus requiring the customer to do no on-line processing during the transaction or during the receipt of electronic goods.
- A variation of the above payment scheme which allows the customer to be totally anonymous from the merchant.

The following section describes passive entities and how they can be used to design electronic payment transaction models. Section 3 presents a new payment scheme which uses the customer as its passive entity. We also present a variation of the payment scheme which offers the customer anonymity. Section 4 provides a comparison with a payment scheme proposed by Kelsey and Schneier [17] which is similar to the new protocol.

2 Passive Entities and Offline Entities

Before describing the use of passive and off-line entities, we must establish some definitions. Electronic payment schemes usually describe a series of messages

which conduct a secure exchange. These messages can be divided into stages. Each of these stages is specifically designed to achieve certain objectives. At least two of the customer, merchant and bank entities within the transaction model interact during each phase. In this paper we recognize that most payment schemes have five phases. These are the registration phase, the withdrawal phase, the negotiation phase, the payment phase, the goods delivery phase and the deposit phase. Not all of these phases are required in all payment schemes but the phase which is mandatory for all schemes is the payment phase. The registration phase allows an entity to register with a trusted third party. The withdrawal phase allows a customer to withdraw electronic funds from the trusted bank entity. The payment phase sees the merchant and the customer exchange electronic funds. The deposit phase consists of the merchant returning the funds to the bank. The negotiation phase is where the merchant and the customer determine the required value to be exchanged in the payment phase. The goods delivery phase is where the merchant delivers the requested goods or services to the customer. The negotiation and goods delivery phases are often left out of payment scheme descriptions because no funds are transferred. The negotiation phase is assumed to have occurred in some manner prior to the commencement of the protocol and the goods delivery phase is assumed to occur in some manner after the completion of the protocol. The importance of the negotiation phase is being seen in electronic auction schemes [12,19].

Within each of the payment scheme phases, each of the entities can be observed to exhibit a certain state of activity. These states are active, off-line or passive. Entities change states from phase to phase. Active entities in one phase are often off-line or passive in other protocol phases.

Active or on-line entities are parties which are actively involved in the payment scheme phase. They must create, sign or encrypt messages and transmit them to other entities to complete the protocol phase. The first electronic payment schemes required that all of the participants be on-line in all phases of the protocol. Designers realized that this would not be suitable for a system which would need to handle millions of transactions per day.

To increase the efficiency of anonymous payment, off-line schemes such as the “cash protocols” by Brands [3] and Ferguson [10] were designed. These payment schemes isolate the bank entity during the payment phase of the protocol. No messages are received or sent to the bank during this time. Much of the complexity of these schemes can be attributed to ensuring that neither the customer nor the merchant cheat while the bank is off-line. Off-line entities have no communication with the other entities. They don’t contribute anything to the completion of the transaction during a protocol phase.

Passive entities are different from off-line entities in that they may transmit and send messages but they do not perform any processing during a protocol phase. Processing in this situation can be defined as any kind of processor intensive operation. These would include any cryptographic encryption, decryption, one-way hash calculation, signature generation or signature verification. It is obvious from this definition that off-line entities are a subset of passive entities.

This is a useful concept because it allows protocol designers to determine the level of processing activity which may be allocated to an entity during a particular phase of the payment scheme.

The payment phase is common to all payment schemes. It can be defined as the part of the payment scheme in which the last of the participants agrees to the transaction. That party usually provides proof of this agreement in the form of a digital signature. With most payment transaction models there are three parties involved in the transaction and all must digitally sign a commitment to complete a transaction. This means that two of the participants may sign the commitment prior to the final participant. Passive entities are usually the first or the second participant in the transaction model to sign the commitment. One of the advantages of using a passive entity is that there is no set time period between the second and third entity signing the commitment. Because the passive entity is not required to conduct any cryptographic processing during its passive phase, an untrusted third party may be used during that phase. The commitment may be posted on an untrusted server's web page or FTP site until the third entity signs it.

In summary, the passive entity has the following properties which define it.

- The passive entity does not conduct any cryptographic processing during its passive stage of the protocol.
- The passive entity may participate in the protocol by transmitting and receiving messages. If a passive entity does not communicate with other parties during a protocol phase it is an off-line entity.
- The passive entity must have given consent to the transaction prior to the payment phase of the exchange.

Most efficient payment schemes have some form of passive entity. Table 1 shows the state of activity for the customer, merchant and bank entities during the payment and negotiation phases of various payment schemes. The negotiation phase has been included because it is also common to many protocols.

Currently most electronic payment schemes which contain a passive entity in the payment phase have assigned that role to the bank. This is mainly because the bank's trusted server is seen to be the most processor and traffic intensive entity of all the payment participants. In Brands [3] Internet cash scheme the bank signs the electronic coin before sending it to the customer. During the payment phase of the protocol only the customer and the merchant are required to sign the coin. The bank is only required again during the deposit when the merchant returns the coins to the bank for verification.

The voucher scheme [11] assigns the passive role to the merchant instead of the bank. In the voucher scheme, the merchant and the bank interact to create a voucher. During this stage both of these entities are required to sign the voucher. This voucher resides on the merchant until an interested customer decides to purchase goods or services from the merchant. The customer downloads these from the merchant in an embedded form with the voucher. But the merchant is not required to take any action. It is passive during the exchange. The customer

Table 1. A comparison of the activity state for various payment schemes during the negotiation and payment phases.

Protocol	Phase	Bank	Customer	Merchant
Brands Internet Cash [3]	Negotiation	Off-line	Active	Passive
	Payment	Off-line	Active	Active
NetBill [8] and SET [14]	Negotiation	Off-line	Active	Passive
	Payment	Active	Active	Active
Vouchers [11]	Negotiation	Off-line	Active	Passive
	Payment	Active	Active	Off-line
Kelsey and Schneier Purchase Orders [17]	Negotiation	Active	Off-line	Active
	Payment	Active	Off-line	Active
Proposed Scheme	Negotiation	Off-line	Passive	Active
	Payment	Active	Passive	Active

then redeems her voucher with the bank's help. One of the advantages of the voucher scheme is that the merchant is basically passive through the whole payment except for the creation of the voucher.

Kelsey and Schneier's [17] "buyer-driven" system contains an off-line customer entity. This system was not designed specifically employing the passive property of the customer. As a result, the bank is required to be on-line during the price negotiation. The customer must also register with the bank for each new purchase order she wishes to create. Their system does not incorporate the delivery of electronic goods or the conduct of any dispute resolution which our scheme does.

The following purchase order payment scheme assigns the customer as a passive entity during the payment phase of the scheme. Like Kelsey and Schneier [17], the bank is required to be on-line during the transaction but the customer does not need to register for each payment.

3 The New Purchase Order Payment Scheme

The following notation is used to denote cryptographic operations. X and Y always represent communicating parties. K always represents a cipher key. When describing the following protocols, the sequence of messages is exchanged among three parties: C , the customer, M , the merchant; and B the bank, acquirer or notary entity.

i. $X \rightarrow Y : Message$ *Message* has been transmitted by entity X to be received by entity Y . This message is the i th message in the protocol.

$E_K(Message)$ *Message*, encrypted by using key K . Any suitably secure algorithm may be used. It is assumed that any shared session keys or public keys have been distributed securely.

	If key K has the form XY , this indicates that the key is known only by X and Y and thus only these entities know the contents of $Message$.
$Sig_X(Message)$	This indicates that $Message$ has been digitally signed by X using public key cryptography. This implies that X 's public key is used to ensure that the message was transmitted by X . It is assumed that the message is transmitted with the signature attached.
$H(Message)$	A cryptographic checksum of $Message$, using an algorithm such as the Secure Hash Algorithm (SHA) one-way hash function.
$Message$	$Message$ in lower case indicates a concatenation of several data items specified in each individual protocol.
$MESSAGE$	$MESSAGE$ in upper case indicates a single data item. Data items separated by commas are concatenated together.

3.1 The Registration Phase

The aim of this phase is to allow the bank server to distribute secret keys to both the customer and the merchant which they will use later in the protocol. It is assumed that the bank and customer have exchanged public keys and the bank and the merchant have also exchanged public keys. Thus the bank can verify the authenticity of both the customer's and the merchant's digital signature. The customer and merchant can also verify the authenticity of the bank's digital signature.

1. $C \rightarrow B : Sig_C(CID, E_{CB}(CustomerAccountDetails))$
2. $B \rightarrow C : Sig_B(E_{CB}(K_C), Expiry_C)$
3. $M \rightarrow B : Sig_M(MID, E_{MB}(MerchantAccountDetails))$
4. $B \rightarrow M : Sig_B(E_{MB}(K_M), Expiry_M)$

In message 1 the customer makes a request to the bank for a key which will allow her to create purchase orders. The customer must provide a unique customer identification number, CID, as well as her account details. The bank saves this information so that he knows which account to debit when merchants offer to fill in the customer's request. By signing the message the customer ensures that no one else can request a customer key in their name. Of course the customer account details must not be allowed to be sent across an open network in the clear.

The bank returns an encrypted customer key, K_C to the customer in message 2. The bank includes an expiry date for the customer key in the message to indicate the lifetime of the key. As the customer key is the key that the customer

uses to create a purchase order, it is essential that it be encrypted so that it is only revealed to the customer. This message must be signed by the bank so that the customer is assured that the customer key is a legitimate key and not a false key from an external party.

Message 3 is very similar to message 1 except that this time the merchant requests a merchant key from the bank. The key will allow him to accept offers made by customers. Like message 1 the merchant must include a unique merchant identification, MID, as well as his encrypted account details. These details tell the bank which account it should deduct when the merchant fills an order.

In message 4, like in message 2, the bank returns an encrypted key. This time to the merchant. The merchant key, K_M enables the merchant to fill customer requests and like the customer key it also has an expiry date. It is best that this message is also signed by the bank to ensure the merchant that he has not received a false key.

It should be obvious that the two messages required to register the customer and the two messages required to register the merchant need not be exchanged directly before a transaction. It is assumed that they are performed well before hand. It is also not necessary that the merchant register after the customer.

Once the customer has successfully received her K_C from the bank she can create an order for each of her required purchases. The customer's purchase order will contain a description of the goods which she requires and a description of the price she is willing to pay for them and an expiry date indicating how long she is willing to wait for an offer. The first step in creating an order requires the customer to generate an order key K_O . The main purpose of this key is to provide a way for the bank to verify that the customer has been previously been approved. This is done in the following manner:

$$K_O = H(K_C, CID, OID, Expiry_O)$$

The order's key contains K_C which is provided by the bank and only known to the customer and the bank. CID is the customer's unique identity which the bank also knows. OID is used to identify this particular order from the customers other orders. Both the CID and the OID are used to provide a unique identifier for the order which cannot be tampered with. An expiry date is included to ensure that the order does not stand for a long period of time without being filled. It is assumed that the hash function is such that the value of K_C can not be discovered even if multiple values of K_O , CID , OID and $Expiry_O$ are known. The use of the key K_O is described in later sections.

One of the advantages of this purchase order scheme is that the customer can create as many orders as she wishes using the same value K_C . This means that she only has to contact the bank once thus saving both her and the bank additional processing and message transfer. It is advisable that the customer periodically to renew K_C for security reasons.

3.2 The Order Posting Phase

During this phase the customer posts her order onto her web page or FTP site, making it available for all merchants to consider. It is not necessary for the merchant to download the purchase order to browse the conditions. It is quite feasible that a customer would advertise the purchase order's conditions on her web site and make available a separate link to download the purchase order.

It is necessary that the customer sign the purchase order even though the bank can determine which customer it is from the CID. Signing the order makes it hard for external parties to alter the contents of the order and it links the order identity and the customer identity to the conditions.

$$5. C \rightarrow M : \text{Sig}_C(K_O, CID, OID, \text{Expiry}_O, \text{Conditions})$$

The purchase order contains a set of humanly readable *Conditions* that the customer wishes to be filled. These conditions should precisely describe the goods which the customer requires and the price which she is willing to pay for them. The purchase order also contains the customer identity and the order identity. These are required to uniquely identify the customer's order. The expiry date of the order is included to indicate to the merchants how long the customer is willing to wait for her order to be filled.

The merchant can download the purchase order with no obligation to fill it. It may be that the merchant does not currently have the goods or may be hoping to find a better offer from another customer but will settle for this offer if none arises. Because the customer has signed the purchase order she has committed to it. The merchant can be assured that the customer will pay for the goods provided.

3.3 The Order Filling Phase

Once the merchant has decided that he will fill a particular order he must download it and conduct the order filling phase of the protocol.

Firstly the merchant must create a purchase key, K_P to encrypt the electronic goods which fulfill the purchase order. This is done in the following manner:

$$K_P = H(K_M, K_O, MID)$$

K_M is the key which the merchant shares with the bank. K_O is required to identify the purchase goods with the customer's purchase request. The merchant's identity is included to provide a way to associate the purchase key with the merchant. Like the order key created by the customer, it is assumed that the hash function used to create K_P is such that K_M cannot be determined even if multiple values of K_P , K_O and MID are known.

6. $M \rightarrow B : \text{Sig}_M(E_{K_P}(\text{Goods}), MID, \text{Sig}_C(K_O, CID, OID, \text{Expiry}_O, \text{Conditions}))$
7. $B \rightarrow M : E_{MB}(\text{Sig}_B(\text{SuccessFlag}))$
8. $B \rightarrow C : E_{CB}(K_P), \text{Sig}_M(E_{K_P}(\text{Goods}), MID, \text{Sig}_C(K_O, CID, OID, \text{Expiry}_O, \text{Conditions}))$

Now that the merchant has generated K_P he must encrypt the requested goods with it. In message 6 of the protocol the merchant signs the goods, his merchant identity and the customer's purchase order which he is filling and sends it to the bank. When the merchant signs this message he is committing the transaction. The customer is assured that the merchant will fill her purchase order.

When the bank receives message 6, it must check if the transaction should go ahead. First of all the bank checks that the customer is a valid customer by verifying that the key K_O has been correctly generated. The bank generates the key using CID , OID and $Expiry_O$ from the message and the original key K_C associated with the CID . The bank checks that the purchase order has not been previously filled. The bank only needs to keep an order's records until the $Expiry_O$ date. The bank also verifies that the customer can pay for the goods provided by the merchant. If it is in the conditions of the transaction the bank can verify that the goods provided by the merchant do fulfill the purchase order. See the disputes resolution section for a description of how the bank does this.

In practice it is unlikely that the bank would be requested to verify the goods during the transaction as it would be time consuming and most likely unnecessary if most merchants are honest. At this point the bank can also deduct funds from both the merchant and the customer for using its facilities.

The bank now sends a *SuccessFlag* to the merchant if it approves of the transaction. This flag must be signed by the bank so that the merchant knows it is a legitimate response. A fake response may fool the merchant into thinking he has sold his goods falsely or that he has failed to sell his goods at all. It is optional to encrypt this message as it only prevents external parties from determining a merchant's sales figures.

It is possible for a merchant to receive a *SuccessFlag* indicating a failed transaction. A transaction is most likely to fail because another merchant may have previously filled the customer's purchase order and that the customer has not yet had a chance to remove the purchase order from its post.

As well as sending the *SuccessFlag* to the merchant in message 7, the bank has the duty of passing on the encrypted goods to the customer in message 8. This message is the same as the message sent to the bank in message 6 except that the release key for the goods K_P is also sent to the customer. This is sent encrypted of course!

3.4 A Variation

One of the disadvantages of the previously described protocol is that the customer does not remain anonymous from the merchant once she posts her requests. Another disadvantage is that the merchant may not be able to complete a transaction because the customer's order may have already been filled, thus potentially wasting the merchant's time and processing.

These problems can be solved if the customer is able to post her purchase order at the bank or some other trusted third party. A limited form of anonymity is achieved if the bank hides the customer's identity. As with other electronic

payment schemes there is a tradeoff between customer anonymity and efficiency. By having a list of customers' orders displayed at the bank, the merchant is only required to access one web page or FTP site. The bank can also keep an accurate list of orders, displaying only unfilled orders. This ensures that merchants fill only unfilled orders. Unfortunately the customer's identity is still known to the bank or trusted third party.

In this variation of the payment scheme the customer does not advertise her own purchase requests. Instead she sends the purchase request to the bank or trusted third party which knows the customer key K_C . It is up to this party to generate an anonymous purchase order. But first the bank must check that this is a legitimate purchase order by verifying the customer's signature (and thus the customer's commitment) and that the key K_O has been created correctly.

$$5. C \rightarrow B : \text{Sig}_C(K_O, CID, OID, \text{Expiry}_O, \text{Conditions})$$

The anonymous purchase order should contain K_O , Expiry_O and *Conditions*. It is then signed by the bank and posted for merchants to view. The bank's signature is a guarantee that a legitimate customer has committed to the request. The bank must keep an index of which K_O matches each CID and OID pair. If a merchant decides to fill the purchase order he must download the anonymous request.

$$5a. B \rightarrow M : \text{Sig}_B(K_O, \text{Expiry}_O, \text{Conditions})$$

To complete the transaction the merchant and bank transmit the same messages as in the original protocol except that in message 6 of the protocol the merchant returns the anonymous purchase order instead of the customer's. After the bank receives message 6 it can reference which customer and order identify the merchant has offered to fill. Thus ensuring that the correct customer receives the merchant's goods in message 8 of the protocol.

To achieve customer anonymity an extra message has been added to the protocol. The bank also has to perform an additional signature operation as well as keep a track of the true customer and order identities of the anonymous purchase orders. At no time during the protocol does the merchant and the customer exchange messages. This affords total anonymity as even router records cannot be checked.

3.5 Protocol Security

The security of this protocol is breached when a transaction is prevented from being completed successfully. Threats to electronic payment scheme security can either be external or internal.

External threats are generated from parties who are external from the parties involved in the protocol. The contents of the messages and the cryptographic tools used to create the messages are specifically used to foil certain external threats. Each message is uniquely identified by CID , MID , and OID to prevent replay attacks. As well as providing non-repudiation, signed messages are

used to prevent external parties from masquerading as participants. Signed messages and the secret customer key K_C , and merchant key K_M help to maintain message integrity and prevent man-in-the-middle attacks. Critical data, such as the purchase key K_P and *SuccessFlag*, are encrypted so that external parties cannot take advantage of the protocol.

Internal threats are generated from participants of a secure protocol. It is possible that either the customer or the merchant will try to cheat the other parties in the transaction. A special dispute resolution protocol is available to deal with these situations.

Dispute Resolution. A dispute occurs if the customer or the merchant is not satisfied that the transaction was completed successfully. The bank is assumed to be an impartial judge trusted by the customer and merchant such that they will abide by all the bank's decisions. It is unlikely that this protocol will be called on frequently as most merchants and customers are assumed to be honest.

The main aim of the dispute resolution protocol is to ensure that the transaction is completed successfully. The dispute resolution protocol consists of the following message from the customer to the bank:

1. $C \rightarrow B : E_{CB}(K_P), \text{Sig}_M(E_{K_P}(\text{Goods}), MID, \text{Sig}_C(K_O, CID, OID, \text{Expiry}_O, \text{Conditions}))$

The customer returns the message it received in message 8 of the protocol. It is now up to the bank to determine who is at fault in the dispute.

Firstly, the bank re-generates K_P using K_O , MID and K_M . The new K_P is compared with the one sent with the message. If it is not the same either the merchant sent an incorrect key or the customer has returned an incorrect K_P in the dispute resolution protocol. In either case the bank will return a correct K_P key (encrypted of course) to the customer.

If K_P is correct the goods sent by the merchant are verified. The bank can use the newly created K_P to decrypt the electronic goods and manually verify that their contents match the description in the customer's purchase order. If the goods do not match the description appropriate action can be started against the merchant who has cheated. Unless the customer is able to fake the merchant's signature, the customer is unable to plant false goods to frame the merchant. The bank has enough information to roll back the transaction.

The merchant or customer cannot deny that they have agreed to the conditions as they both have signed the purchase order.

3.6 Comparisons with Kelsey and Schneier

The original purchase order protocol developed by Kelsey and Schneier [17] was not developed using the concept of a passive entity. Instead they sought to design a buyer driven payment scheme. We choose the Kelsey and Schneier protocol as a benchmark because their "buyer-driven" design framework is the most similar

to ours. In fairness, it should be noted that Kelsey and Schneier provide services such as anonymity which are not provided by our scheme.

Kelsey and Schneier’s protocol uses four entities to conduct a transaction: A customer, a merchant, an arbiter and a server. The customer registers with an arbiter to get permission to post her purchase order. Once the customer has the arbiter’s acceptance she can then post her purchase order on a server. A customer must get an arbiter’s acceptance for each order she wishes to post. Merchants must also get permission to browse purchase order’s from the server. The browsing involves the merchant using a special key to authenticate each purchase order. The merchant must request each purchase order from the server separately even if he wishes to fill it or not. Once a merchant decides to accept an order it is bound to him with the assistance of the server. It is up to the merchant to now deliver his goods as described in the purchase order’s conditions.

One of the differences in Kelsey and Schneier is that there is no set method to ensure that the merchant delivers his goods. The merchant can simply refuse to do so and the customer has no way of knowing that her offer was accepted.

Table 2. A processing and message transmission comparison of Kelsey and Schneier’s scheme with the proposed payment scheme.

	Kelsey and Schneier Purchase Orders	Proposed Payment Scheme
On-line Messages Transmitted	14 + number of Orders	4
Off-line Messages Transmitted	14 + number of Orders	4
Total Messages Transmitted	14 + number of Orders	8
Customer Encryptions (On-line)	4	1(0)
Merchant Encryptions (On-line)	3	2(1)
Arbiter/Server/Bank Encryptions (On-line)	7 + number of Orders	4(2)
Total Encryptions (On-line)	14 + number of Orders	7(3)
Customer Hash Calculations	2	1
Merchant Hash Calculations	2	1
Arbiter/Server/Bank Hash Calculations	7	2
Total Hash Calculations	11	4
Customer Signatures (On-line)	4	2(0)
Merchant Signatures (On-line)	3	2(1)
Arbiter/Server/Bank Signatures (On-line)	7 + number of Orders	3(1)
Total Signatures (On-line)	14 + number of Orders	7(2)

In terms of processing and message transmissions, the current payment protocol is more efficient. In Table 2 we provide a comparison of our payment schemes. Several protocol design issues decrease the efficiency of Kelsey and Schneier. All messages must be transmitted for the payment to be completed. Whereas in the new protocol only four messages need to be sent for a payment to be completed.

The remaining four messages in the new protocol only need to be performed once or at periodic intervals. Kelsey and Schneier require the bank to be on-line while a merchant browses the orders. This greatly increases the traffic generated by the bank.

Because of the smaller number of messages required, less encryptions and signature generation is required. Kelsey and Schneier have blanket encryptions and signatures for all messages to ensure the security of their scheme. This policy further increases the bank's processing load. Each message containing a single customer order that is sent to browsing merchants requires a signature and encryption. Overall each participant must use more processing to complete a transaction.

Customer anonymity is the main advantage that Kelsey and Schneier do have over our original protocol. A limited level of anonymity can be achieved in the new purchase order protocol using a variation of the protocol presented in section 3.4.

One of the disadvantages of designing protocols with specific passive entities is that the other participants in the protocol become more loaded. In this case, the bank must do the bulk of the processing in terms of encryption and signature generation for each transaction. The bank must also store a record of all transactions until the expiry date for repudiation reasons. This may not be suitable for some payment implementations.

4 Conclusion

This paper describes a concept for designing new transaction models for electronic payment schemes. Passive entities can be used to develop transaction models that result in electronic payment systems which may be more efficient and elegant than the traditional credit card model. This paper also presents a new payment scheme designed using this concept. The proposed scheme, while providing the same security, is more efficient than other similar schemes.

References

1. M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, pages 122-136, May 1994.
2. Mihir Bellare, Juan A. Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Micheal Steiner, Gene Tsudik, and Micheal Waidner. iKP – A Family of Secure Electronic Payment Protocols. In *Proceedings of the First Usenix Workshop on Electronic Commerce*, New York, July 1995.
3. Stefan Brands. Untraceable Off-line Cash in Wallets with Observers. In *Advances in Cryptology - Proceedings of CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302-318. Springer-Verlag, 1993.
4. Stefan Brands. Electronic Cash on the Internet. In *In Proceedings of the Internet Society 1995 Symposium on Network and Distributed System Security*, pages 64-84, 1995.

5. Jan Camenisch, Ueli Maurer, and Markus Stadler. Digital Payment Systems with Passive Anonymity-Revoking Trustees. In *Proceedings of ESORICS '96*, volume 1146 of *Lecture Notes in Computer Science*, pages 33–34. Springer-Verlag, 1996.
6. Agnes Chan, Yair Frankel, and Yiannis Tsionis. Easy Come - Easy Go Divisible Cash. In *Advances in Cryptology - Proceedings of EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer-Verlag, May 1998.
7. David Chaum, Amos Fiat, and Moni Naor. Untraceable Electronic Cash. In *Advances in Cryptology - Proceedings of CRYPTO '88*, *Lecture Notes in Computer Science*, pages 319–327. Springer-Verlag, 1990.
8. Benjamin Cox, J. D. Tygar, and Marvin Sirbu. NetBill Security and Transaction Protocol. In *Proceedings of the First Usenix Workshop on Electronic Commerce*, New York, July 1995.
9. Tony Eng and Tatsuaki Okamoto. Single-Term Divisible Electronic Coins. In *Advances in Cryptology - Proceedings of EUROCRYPT '94*, number 950 in *Lecture Notes in Computer Science*, pages 306–319. Springer-Verlag, 1994.
10. Niels Ferguson. Extensions of Single Term Coins. In *Advances in Cryptology - Proceedings of CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 292–301. Springer-Verlag, 1993.
11. Ernest Foo and Colin Boyd. A Payment Scheme Using Vouchers. In *Financial Cryptography '98*, number 1465 in *Lecture Notes in Computer Science*, pages 103–121. Springer-Verlag, Feb 1998.
12. M. K. Franklin and M. K. Reiter. The Design and Implementation of a Secure Auction Service. In *IEEE Transactions on Software Engineering*, volume 22(5), pages 302–312, May 1996.
13. A. O. Freier, P. L. Karlton, and P. C. Kocher. The SSL Protocol Version 3.0. <http://www.netscape.com/eng/ssl3/>, 1996.
14. VISA/MasterCard International. Secure Electronic Transaction (SET) Specification Book 1: Business Description. http://www.setco.org/set_specifications.html, 1997.
15. VISA/MasterCard International. Secure Electronic Transaction (SET) Specification Book 2: Programmer's Guide. http://www.setco.org/set_specifications.html, 1997.
16. VISA/MasterCard International. Secure Electronic Transaction (SET) Specification Book 3: Formal Protocol Definition. http://www.setco.org/set_specifications.html, 1997.
17. John Kelsey and Bruce Schneier. Conditional purchase orders. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*. ACM Press, 1997.
18. B. Schoenmakers. Basic Security of the eCash(tm) Payment System. In *State of the Art in Applied Cryptography - Course on Computer Security and Industrial Cryptography*, volume 1528 of *Lecture Notes in Computer Science*. Springer-Verlag, Jun 1997.
19. S. Stubblebine and P. Syverson. Fair On-line Auctions without Special Trusted Parties. In *Proceedings of Financial Cryptography 99*, *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

Key Recovery System for the Commercial Environment

Juanma González Nieto, Kapali Viswanathan, Colin Boyd, and Ed Dawson

Information Security Research Centre
Queensland University of Technology
GPO Box 2434, Brisbane 4000, Australia.
{gonzalez, viswana, boyd, dawson}@fit.qut.edu.au

Abstract. We propose an efficient, scalable, flexible key recovery solution for use in commercial environments. Our scheme takes a new approach in the design of key recovery systems, called hybrid key escrow. We shall demonstrate the claims by comparing the computation and communication requirements for our proposal with the key recovery solution implemented by IBM.

Keywords: Hybrid key escrow, commercial key recovery, key escrow, key management, software key recovery.

1 Introduction

Cryptography protects the confidentiality of information by limiting access to the 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. This usually involves a Trusted Third Party (TTP) that has the capability of restoring the appropriate decryption keys. 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 and forensic analysis when strong cryptography is used.

Key recovery has been the subject of much research during the last few years [5]. However it has been mainly driven by clearly defined government requirements [18], [9], [6] and has led to rather unsatisfactory proposals such as

the renowned Clipper Chip in the USA and the GCHQ protocol in the UK [21][4]. Only recently, serious attempts to understand the requirements for key recovery in the commercial sector have been undertaken [15], [12] and work on the formulation of key recovery standards is still being done [17].

Many of the government requirements that have shaped the designs of KR schemes, can be relaxed when translated to the commercial arena. This greatly simplifies the design of KR schemes that are suitable for commercial environments. The main differences between commercial and government requirements can be summarised as follows:

- Scope of the KR scheme - Whereas governments desire an ubiquitous deployment of KR that extends multiple domains, applications and communication layers in which encryption is used for confidentiality purposes, the business scope for KR is much more localised and discriminatory in its applications. Thus for instance, many commercial organisations do not require recoverability of encrypted communications.
- Enforceability - From a law enforcement viewpoint, it is strongly required that the users of a cryptographic system cannot circumvent the KR mechanism (KRM)[9]. This however is not generally the case in commercial environments, specially when the primary motivation for KR is just the back-up of keys for self-recovery. Even when KR is deployed to satisfy a requirement for the monitoring of encrypted communications in a corporate organisation, the enforceability of the KR mechanism can be based on the physical control that the organisation has on the computing equipment and on the applications that can be run on them, as well as on random checks to verify that the encrypted information is recoverable as per security policy. Clearly, such means of enforceability become impractical in a global KR infrastructure.

In this paper we first give an overview of existing KR techniques, and point out some of their advantages and disadvantages. We discuss the business requirements for KR, as well as the suitability of existing KR techniques. We realise that the main shortcoming of the existing proposals in order to meet such requirements, is their poor efficiency, and subsequently propose a new scheme that significantly reduces the computational and bandwidth overhead when compared with existing schemes. This is exemplified by a comparison between our new proposal and the prominent scheme of Gennaro *et al.* [8], which has been integrated into the IBM's KeyWorks product line [19].

2 Key Recovery Techniques

In this paper we use the following terminology:

Cryptographic End Systems (CES). These provide the functionality that allows cryptographic operations. CES are implemented as a product in software and/or hardware. KR is only concerned with those CES that implement confidentiality services.

End Users (EU). Entities that employ CES for secure data communications and storage. End users may or may not be part of a public key infrastructure (PKI), i.e. be in possession of valid public key certificates. Throughout this paper, we use the term “user” to refer to either CES or EU.

Key Recovery Agents (KRA). Trusted third parties that perform KR in response to an authorised request.

Key Recovery Requesters (KRR). Authorised entities that request KR from the KRA. The KRR could be for instance a user that needs extraordinary access to her own data, an auditor, etc. It can also refer to a third party that acts as a proxy between them and the KRA.

Key Recovery Information (KRI). Aggregate of data that is needed by the KRA in order to complete a KR request, eg. a session key encrypted under the KRA’s public key.

For simplicity, in what follows we describe KRMs in a communications scenario between two users that participate in a KR infrastructure. Notice that storage applications can be considered as a special case of communication in which both the sender and the receiver are the same entity.

KR techniques are commonly categorised into two types: key escrow and key encapsulation [17].

Key escrow encompasses those KRMs in which KRAs directly store keys, or information related to the keys of the users of the system. Thus, provided that users employ such keys for direct bulk data or session key encryption, the KRAs are automatically given access to the plaintext of communications. In order for the KRM to work, users are restricted on the usage of the escrowed keys to those key establishment mechanisms that allow KRA access to the session key. This not only reduces the cryptographic flexibility of the system, but also is the main hurdle for those applications of key recovery that try to enforce the KRM by purely cryptographic means[13]. Controlling the usage of keys is made even more difficult by the fact that there are legal issues that disallow the escrowing of keys used for non-repudiation. Key escrow techniques include [11,23].

Key encapsulation is also known as *virtual addressing* [14]. KR is achieved by encrypting the established session keys for each KRA that requires access to them. These encrypted (encapsulated) keys are sent together with the encrypted messages. An advantage of key encapsulation mechanisms is that they do not require users to register with the KR infrastructure. As well, no restriction on the key management techniques used to directly support the confidentiality service is imposed on them. However, a significant computational and communication overhead is introduced. Examples of key encapsulation techniques can be found in [8,22]. As with key escrow techniques, enforcing KR by cryptographic means only, appears to be extremely difficult. In [22] an ElGamal based key encapsulation scheme is proposed, in which the sender provides a publicly verifiable proof that the same session key has been encrypted for the receiver and the corresponding KRAs. However, attacks have been described in [20]. Furthermore, the type of attack described in [20] seems to affect all KR techniques.

3 Requirements for a Commercial KR System

The primary goal of our research is to design a product that satisfies the business needs for key recovery. For this purpose, the first step towards an appropriate key recovery scheme is to clearly understand the business requirements. In this paper, we concentrate only on those requirements which influence the design of the key recovery protocols. This approach allows us to concentrate on the mechanism for key recovery during the system design and isolates the myriad of policy issues that the mechanism could be subjected to.

In what follows we discuss the main requirements that we incorporate in our design.

Flexibility:

R 1: *The key recovery mechanism must support data communications and storage applications.* Although as already mentioned, the main use of KR in corporate organisations is intended to be storage applications, we pursue the design of a KRM that can also be efficiently deployed for data communications.

R 2: *The key recovery mechanism shall be independent of the cryptographic algorithm used for confidentiality.* The rationale for this requirement is twofold. Firstly, it is considered to be a good security practice to design cryptographic protocols independent of the underlying algorithms. Secondly, two communicating entities employing a common protocol but different sets of cryptographic algorithms can agree upon a common set of algorithms to carry out the protocol. This will greatly increase the inter-operability between domains with varying policies.

R 3: *The key recovery mechanism shall be independent of the key management and distribution mechanisms that support the confidentiality service.* Requirement **R 3** has a similar effect as **R 2**, except that now inter-operability between different protocol logics is the focus.

Enforceability:

R 4: *Key recovery mechanism shall prohibit single rogue users from subverting the KRM.* By incorporating this requirement we aim at complying with the enforceability requirement defined in [17] for a Level-2 (high security) KR product. This requirement can be satisfied if a mechanism is provided that allows the receiving end of a communication association to check whether KR by the appropriate KRAs is enabled. Notice that this level of enforceability is significantly lower than the one that has been traditionally required for law enforcement KR, and easier to implement. Requirement **R 4** does not preclude two cooperating rogue users circumventing the KR mechanism while at the same time communicating securely.

Efficiency:

R 5: *The computational, communicational and infra-structural overhead incurred by the addition of a key recovery scheme shall be kept to a minimum.* This requirement emphasises the importance of efficiency, when employing current computing and communication technology, in the design of key recovery systems, specially if intended for real-time communications.

Unobtrusiveness:

R 6: *Key recovery mechanisms should affect confidentiality services and should not affect other services.* Requirement **R 6** defines the precise role of a key recovery system in a domain. It must have the capability to *revoke* confidentiality and must not alter the security guarantees of other cryptographic services like authentication.

R 7: *The key recovery mechanism shall not reduce the effective strength of the encryption.* In order to satisfy this requirement, the key recovery system must not leak any useful session key information, except obviously to the appropriate KRAs.

Fine Granularity:

R 8: *The number of session keys that can be recovered from a single instance of KRI shall be small enough so as to ensure fine granularity of authorised interception periods.* In order to satisfy this requirement, KR that have been authorised for a specified encrypted data or time period should not compromise data outside the scope of the authorisation.

Scope:

R 9: *The key recovery mechanism shall be scalable so as to accommodate big multinational corporate organisations, and multiple policy domains.*

Dispersion:

R 10: *The key recovery mechanism shall allow the distribution of the KRA function among multiple entities so as to require their joint cooperation in order to perform key recovery.*

4 KR Mechanism Design

Once the requirements for our design have been established, we need to evaluate the suitability of existing KRMs. Those KRMs based on the key escrow technique, as defined in Section 2, clearly violate the cryptographic flexibility requirement that derives from **R 2** and **R 3**, since, as we explained in Section 2, key escrow schemes limit the choice of key establishment protocols that can be used.

KRMs based on the key encapsulation technique, on the other hand, seem more appropriate in order to satisfy our design requirements. As already mentioned, the major drawback of these KRMs is the computational and bandwidth overhead that they introduced. As an example, a proposal for using key encapsulation mechanisms with IPSec [12] introduces a 30% to 50% bandwidth overhead [7].

Since efficiency is the main obstacle found in key encapsulation schemes towards the satisfaction of our design requirements, we investigated ways of minimising the overhead that such schemes carry with themselves. Our strategy for a new KRM is to combine both key escrow and encapsulation techniques in order

to achieve a computational and communicational efficiency greater than key encapsulation only schemes, while at the same time satisfying all the requirements in Section 3.

The key encapsulation part of our approach is similar to the one taken by Gennaro *et al.* [8], which has been integrated into IBM's KeyWorks product line [10], and which is one of the most efficient key encapsulation mechanisms. However by using a key escrow procedure, we have been able to significantly reduce both the computational load and the communications overhead.

4.1 The New Key Recovery System

In this section we describe in detail our new proposal for a KR scheme, followed, in the next section, by a comparison between the new scheme and the one proposed in [8].

Overview. The protocol has three phases, as follows:

1. **Registration Phase:** Initially users register with one or more KRAs in their domain. During this phase each user establishes a Diffie-Hellman (DH) shared secret with each KRA.
2. **Communication Phase:** This phase corresponds to the communication of encrypted data and to the generation, delivery, and (optionally) validation of the KR fields that enable recovery of the session keys used in encrypting the data. This phase is further subdivided in two:
 - a) *Initial Set-up:* This phase is performed by the two communicating parties the first time they establish a communication association. At this stage the users establish new DH shared secrets from the values already shared with their respective KRAs. This new values can be recomputed collectively by the KRAs in each domain. This phase can be made a one-off process if the established values are cached for reuse in successive sessions; and,
 - b) *Encrypted session:* After the initial set-up, a new key encrypting (KEK) can be generated as a one-way function of the shared secrets established in the previous phase and a random session identifier. The KEK is used to encrypt the session key employed for bulk encryption, which is then appended to the encrypted bulk data.
3. **Key Recovery Phase:** The KRAs associated to each communicating party in each domain can restore the KEK on a per session basis when presented with the appropriate authorised request and KR fields. The release of one or more KEKs does not compromise any other keys used by the users.

KR Protocol Descriptions

System Set-up: The authorities of the system choose a large prime p of the form $p = 2q + 1$, such that q is a sufficiently large prime. The KR protocols use operations in the multiplicative group $\mathbb{Z}_p^*$, with g an element of order q .

For brevity, we restrict our description to a basic scenario in which two users A and B , pertaining to two different domains, establish a communications association. Each user has two associated KRAs: A_1 and A_2 for user A and, B_1 and B_2 for user B . Each KRA, A_i and B_i , chooses a master key MK_{A_i} and MK_{B_i} respectively. It is important to emphasise that this is just one possible operational scenario, chosen so as to facilitate the description and analysis of KR mechanism. The key recovery system is flexible enough to adapt to a wide variety of practical applications. Thus, for example, this basic scenario can easily be extrapolated to any number of KRAs. Furthermore the number of KRAs for each user does not need to be the same.

Registration with KRAs phase: Initially users register with the KRAs within their KR domain. The extension of the KR domain depends on the applicable enterprise policies. For example it may be a branch of a corporate organisation. These policies may mandate that cryptographic end systems be registered. This way, unregistered end users can be permitted the use of CESs and still allow KR.

The goal of this phase is to establish a Diffie-Hellman shared secret between the user and her KRAs. Table 1 shows the protocol message exchange for the registration of A . User A generates a random DH key pair $(x_A, y_A = g^{x_A})$ and sends y_A to each KRA, A_i . Then, A_i generates a new value:

$$a_i = \mathcal{H}_1(g^{x_A}, MK_{A_i}) \quad (1)$$

where $\mathcal{H}_1$ is a keyed hash function that is partial pre-image resistant and strong computation resistant [16, pp. 325], and with the additional property [3] that $(a_i, p - 1) = 1$. The DH secret is computed as:

$$\alpha_{A_i} = (g^{x_A})^{a_i} \quad (2)$$

This minimises the risk of compromise of MK_{A_i} from a user. Note that, in order to obtain MK_{A_i} an attacker has to firstly attack α_{A_i} to find a_i , and then attack the hash function. Each A_i then returns the public KR keys $\gamma_{A_i} = g^{\alpha_{A_i}}$, certified by A_i , and the corresponding g^{a_i} . Before accepting the certificates, A calculates $\alpha_{A_i} = (g^{a_i})^{x_A}$ and checks the received γ_{A_i} values.

As we explain in Section 4.2, depending on the application, this phase can be adapted to be non-interactive. Another possible variation would merge the registration of the user with her Certification Authority (CA) and with the KRAs into a single process, at the end of which the user would obtain a single certificate for her authentication public key, and KR's public keys, thus saving bandwidth resources. Notice however that only the KR key pair is escrowed with the KRAs.

Table 1. KR Registration Phase

User A	KRA A_i
Choose $(x_A \in_R \mathbb{Z}_p^* \mid (x_A, p-1) = 1)$	
$y_A = g^{x_A}$	$\xrightarrow{y_A} y_A^2 \stackrel{?}{\neq} y_A^q \stackrel{?}{\neq} 1$ $a_i = \mathcal{H}_1(y_A, MK_{A_i})$ $\alpha_{A_i} = y_A^{a_i}$ and $\gamma_{A_i} = g^{\alpha_{A_i}}$
$\alpha_{A_i} = (g^{a_i})^{x_A}$	
$\gamma_{A_i} = g^{\alpha_{A_i}}$	
$(g^{a_i})^2 \stackrel{?}{\neq} (g^{a_i})^q \stackrel{?}{\neq} 1$	
$Check(\gamma_{A_i}, cert_{A_i})$	$\xleftarrow{\gamma_{A_i}, cert_{A_i}, g^{a_i}} cert_{A_i} = Certify(y_A, \gamma_{A_i})$ $\xrightarrow{ACK, NACK} Publish(\gamma_{A_i}, cert_{A_i})$

Communication Phase: In this phase users A and B establish an encrypted communications association. Each user holds valid KR certificates of each other with their corresponding public KR keys.

Initial set-up:

User A calculates the following:

$$u_1 = \gamma_{B_1}^{\alpha_{A_1}} \quad (3)$$

$$u_2 = \gamma_{B_2}^{\alpha_{A_2}} \quad (4)$$

The values u_1 and u_2 can be stored for reuse in all subsequent communications between the two users or recalculated at any convenient frequency. Later on, the first time that user B receives an encrypted message from A , user B needs to compute the same values as:

$$u_1 = \gamma_{A_1}^{\alpha_{B_1}} \quad (5)$$

$$u_2 = \gamma_{A_2}^{\alpha_{B_2}} \quad (6)$$

Importantly, this set-up phase requires only two exponentiations. In the general case (see Section 4.2), the number of exponentiations is equal to the maximum number of KRAs of the users. This compares very favourably with similar schemes such as in [8], making the KR quite efficient even in the cases where the u -values are not cached.

Encrypted session:

We assume that the session key K_S which is used for confidentiality is already established and known to both A and B using some external mechanism. The goal of this phase is to prepare the KR fields that will allow the corresponding KRAs access to K_S . These KR fields will also be validated by the receiver B , if mandated by the KR policy. A does as follows:

1. Generate random session identifier ID_S ;
2. compute $KEK_1 = \mathcal{H}_2(u_1, ID_S)$ and $KEK_2 = \mathcal{H}_2(u_2, ID_S)$, where $\mathcal{H}_2$ is a one-way hash function that is pre-image resistant and strong computation resistant;
3. calculate key encrypting key, $KEK = KEK_1 \oplus KEK_2$;
4. encrypt session key using a symmetric cipher, $ESK = \text{symmEnc}(K_S, KEK)$;
5. send $\{ESK, ID_S, \text{cert}_{A_1}, \text{cert}_{A_2}, \text{cert}_{B_1}, \text{cert}_{B_2}\}$ together with encrypted bulk data to B .

When B receives the messages he can validate the KR fields. This could be done for two purposes: to verify that the session key can be recovered by B 's KRAs, and, optionally to verify that KR is also possible by A 's KRAs. For the second case B will have to ensure that A 's KR public keys are authentic, by verifying cert_{A_1} and cert_{A_2} . The validation process is then done as follows:

1. Calculate u_1 and u_2 corresponding to the KR public keys identified in cert_{A_1} and cert_{A_2} , as explained in the previous phase, if not known from previous sessions;
2. compute $KEK_1 = \mathcal{H}_2(u_1, ID_S)$ and $KEK_2 = \mathcal{H}_2(u_2, ID_S)$;
3. calculate key encrypting key, $KEK = KEK_1 \oplus KEK_2$.
4. The validation is successful if the received $ESK = \text{symmEnc}(K_S, KEK)$.

In the case where validation fails a KR policy being enforced by a CES may prohibit decryption of the received data using K_S . Alternatively, the KR policy may allow decryption as long as the receiver generates the KR field himself, and makes them appropriately available. Notice however that most commercial applications will not require a validation function at all.

Key Recovery: During this phase, a key recovery requester (KRR) with the appropriate authorisation in a domain interacts with the KRAs identified in an “intercepted” message in order to recover the session key used in such message.

Figure 2 depicts the interactions between the KRR and the KRAs in A 's domain. KR in B 's domain is done in the same way. Note that, for simplicity, we assume the existence of authenticated channels between the KRR and the KRAs. For each KRA A_i , associated with A , the KRR does as follows:

1. sends the “intercepted” values; $\{ESK, ID_S, \text{cert}_{A_i}, \text{cert}_{B_i}\}$ together with other information such as authorisation for key recovery.
2. computes corresponding u -value by doing:

$$a_i = \mathcal{H}_2(g^{x_A}, MK_{A_i}) \quad (7)$$

$$\alpha_{A_i} = (g^{x_A})^{a_i} \quad (8)$$

$$u_i = y_{B_i}^{\alpha_{A_i}}; \quad (9)$$

3. recomputes the key encrypting key, $KEK_i = f(u_i, ID_S)$; and
4. sends KEK_i to KRR.

Once the KRR obtains both KEK_i values he proceeds as follows:

1. computes the key encrypting key, $KEK = KEK_1 \oplus KEK_2$; and,
2. recovers the session key by decrypting the “intercepted” ESK , $K_S = \text{symmDec}(ESK)$.

Table 2. KR phase in A ’s domain

KRR	KRA A_i
	$\xrightarrow{ID_S, \text{cert}_A, \text{cert}_B} a_i = \mathcal{H}_1(y_A, MK_{A_i})$ $\alpha_{A_i} = y_A^{a_i}$ $u_i = y_{B_i}^{\alpha_{A_i}}$
$KEK = KEK_1 \oplus KEK_2$ $K_S = \text{symmDec}(ESK)$	$\xleftarrow{KEK_i} KEK_i = \mathcal{H}_2(u_i, ID_S)$

4.2 Variations

One of the main objectives in designing our KR system was to provide a scheme flexible enough to adapt to a wide range of operational scenarios. In this section, we explore some of the multiple variations and enhancements that allow the key recovery mechanism to efficiently perform in most practical environments.

Unequal Number of KRAs: In the general case where A has KRAs $A_1, \dots, A_m$; and B has KRAs $B_1, \dots, B_n$, with $m > n$ a possible method to compute the u -values is as follows:

$$\begin{aligned}
 u_1 &= DH(\alpha_{A_1}, \alpha_{B_1}) \\
 u_2 &= DH(\alpha_{A_2}, \alpha_{B_2}) \\
 &\dots \\
 u_n &= DH(\alpha_{A_n}, \alpha_{B_n}) \\
 u_{n+1} &= DH(\alpha_{A_{n+1}}, \alpha_{B_1}) \\
 u_{n+2} &= DH(\alpha_{A_{n+2}}, \alpha_{B_2}) \\
 &\dots \\
 u_m &= DH(\alpha_{A_m}, \alpha_{B_{m-n}})
 \end{aligned}$$

independently of whether A is the sender or the receiver, and where $DH(a, b) = g^{ab}$.

Non-interactive registration of users: Many applications will not require that users be in possession of KR certificates. In particular, this may be the case if no validation of KR fields is needed or if, in a communications scenario, the receiver only validates that KR is possible in his domain. For this situation, users need not register with the KRAs interactively. The registration phase described above can be adapted to this scenario as follows.

Each KRA in the user's domain, say A 's, has a DH key pair, $x_{A_i}, y_{A_i} = g^{x_{A_i}}$. In order to increase the security of the KR system each KRA would actually have many such pairs. User A can establish new α_{A_i} values with the KRAs in a non-interactive fashion by doing the following:

1. Obtain authentic DH public keys for the KRAs, y_{A_i} .
2. Choose random $x_A \in \mathbb{Z}_p^*$ and compute $y_A = g^{x_A}$
3. Calculate DH shared secret key, $\alpha_{A_i} = y_{A_i}^{x_A}$.
4. Compute public KR values of A , $\gamma_{A_i} = g^{\alpha_{A_i}}$

A would use the γ_{A_i} values for the creation of KR fields, and append these values together with y_A to the encrypted data. If the KRAs have multiple DH key pairs, the user would also need to include identifiers of the keys used in computing α_{A_i} . In a communications scenario where A is the receiver, A would also have to authenticate γ_{A_i} as hers to the sender.

5 Efficiency: A Comparative Study

In this section we shall present a comparison between our scheme and the proposal by Gennaro *et al.* [8]. As already mentioned the scheme in [8] utilises a key encapsulation mechanism, in which users do not escrow any long term keying information with the KRAs. As before, let x_A and x_B be the private keys of two communicating parties A and B . Each user belongs to a different domain. Let a_1 and a_2 be the private keys of the KRAs in A 's domain, and similarly b_1 and b_2 the private keys of the KRAs in B 's domain. As in our scheme, the generation of KR information is broken into two phases. During the first phase A and B establish shared secrets with each KRA's. For each pair of users A and B , the shared secrets are computed as follows:

1. $u_{AB} = DH(x_A, x_B)$
2. $u_{AB1} = DH(u_{AB}, a_1)$
3. $u_{AB2} = DH(u_{AB}, a_2)$
4. $u_{BA1} = DH(u_{AB}, b_1)$
5. $u_{BA2} = DH(u_{AB}, b_2)$

A pictorial representation of how the above values are derived is shown in Figure 1. The circles represent the Diffie-Hellman operation, while the squares represent the different participating entities. Notice that the arrows have been drawn dotted to indicate the new values that must be computed for every pair of users. In terms of computational overhead, Gennaro *et al.* scheme requires five modular exponentiations (one for each Diffie-Hellman operation), which are by far the most computationally expensive operations in the protocol. Although these values can be cached for reuse, we have to notice that in many applications it is not practically possible to *securely* store such amount of information for each party that a user communicates with. For example, this would be the case in applications that rely on smart cards for secure storage of sensitive information.

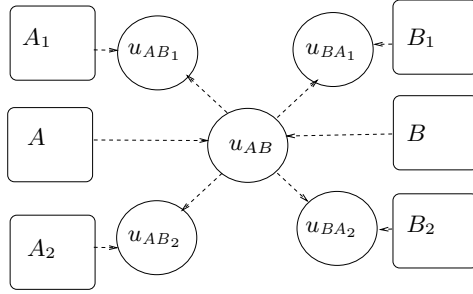


Fig. 1. Gennaro *et al.* key encapsulation scheme

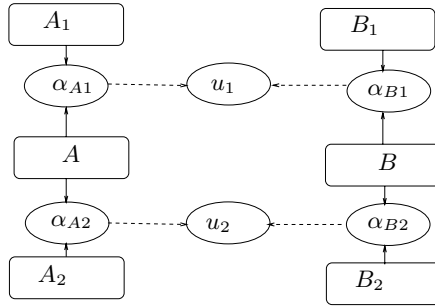


Fig. 2. New hybrid KR scheme

The pictorial representation of our scheme is as shown in Figure 2. By allowing the users to register with their respective KRAs, we reduce the number of Diffie-Hellman operations that need to be computed for each pair of users. Now the α -values remain constant (solid arrows) and only u_1 and u_2 have to be computed.

Table 3. Initial Set-up Phase

	Our proposal	Gennaro <i>et al.</i> [8]
Exponentiations	$\max(n_A, n_B)$	$n_A + n_B + 1$
Symmetric Operations	None	$2(n_A + n_B)$
Hash computations	None	$n_A + n_B$

For a general scenario in which A has n_A KRAs, and B is associated to n_B KRAs Table 3 and Table 4 compare the computational load in both KR schemes. In the tables, $\max$ is a function that returns the maximum value of the two inputs. Typically n_A and n_B are the same value, in which case the savings in exponentiations is more than 50%. This clearly represents a significant reduction on the computational overhead.

Table 4. Encrypted Session Phase

	Our proposal	Gennaro <i>et al.</i> [8]
Symmetric Operations	1	$2(n_A + n_B)$
Hash computations	$\max(n_A, n_B)$	None

With respect to bandwidth we notice, that in our scheme the random seeds that generate the KEKs are calculated directly by the KRAs, and therefore there is no need to send them in every message, as it is the case in Gennaro *et al.* [8] scheme. Also, in a typical scenario where n_A and n_B are approximately the same value, the number of u -values is reduced by more than 50%. Hence, the amount of information that has to be securely cached is also reduced by the same ratio. Finally, in the scheme proposed by Gennaro *et al.*, the session key is encapsulated in separate key recovery fields, encrypted under different KEKs for each domain. Whereas, in our proposal the session key is encapsulated in only one field, which can be accessed by KRA's in both the domains.

6 Conclusion

We presented an efficient, scalable, certification based key recovery system that is a hybrid of key escrow and key encapsulation techniques. The proposed key recovery system is highly flexible and can be employed in a variety of policy environments. It possess all the properties required for commercial key recovery systems.

Additionally, we presented a comparison of our proposal with IBM's key recovery solution and demonstrated that our scheme is more efficient in terms of computation, storage and bandwidth. Thereby, we strongly believe that our scheme is more suitable for low memory, reduced power, distributed systems, such as smart card and mobile instrument based systems, than that of IBM's solution. Due to the resulting increase in efficiency, our system will be more suited to network layer (layer 2 and 3 in the OSI model) key recovery applications such as IPsec [1,2].

References

1. R. Atkinson. IP Authentication Header. RFC 1826, NRL, August 1995.
2. R. Atkinson. IP Encapsulating Security Payload. RFC 1827, NRL, August 1995.
3. Colin Boyd. Enforcing traceability in software. In *Information and Communication Security - First International Conference, ICICS'97*, pages 398–408. Springer, 1997.
4. 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>.

5. D. E. Denning and D. K. Branstad. A taxonomy for key escrow encryption systems. *Communications of ACM*, pages 34–40, March 1996.
6. ETSI TC-STAG. Security Techniques Advisory Group (STAG); Definition of User Requirements for Lawful Interception of telecommunications; Requirements of the Law Enforcement Agencies, December 1996. ETR 331.
7. P.A. Fouque, G. Poupard, and J. Stern. Recovering keys in open networks. In *Proceedings of IEEE Information Theory and Communications Workshop (ITW'99)*. Kruger National Park, SA, June 1999.
8. Rossario Gennaro, Paul Karger, Stephen Matyas, Mohammad Peyravian, Allen Roginsky, David Safford, Michael Willet, and Nev Zunic. Two-phase cryptographic key recovery system. *Computers and Security*, 16:481–506, 1997. Available at: http://www.ibm.com/security/library/wp_key.html.
9. Government Information Technology Services. Key Recovery Demonstration project: Implementation Evaluation Criteria. Available at <http://gits-sec.treas.gov/krdpeval.htm>.
10. IBM KeyWorks Toolkit. Homepage, <http://www-4.ibm.com/software/security/keyworks/>.
11. Nigel Jefferies, Chris Mitchell, and Michael Walker. A proposed architecture for trusted third party services. In *Cryptography: Policy and Algorithms*, pages 98–104, 1996.
12. Key Recovery Alliance. Business Requirements for Key Recovery, white paper, December 1997. Available at <http://www.kra.org>.
13. Lars R. Knudsen and Torben P. Pedersen. On the difficulty of software key escrow. In *Advances in Cryptology - EUROCRYPT'96*, Lecture Notes in Computer Science, pages 1–8. Springer-Verlag, 1996.
14. L.R. Knudsen and K.M. Martin. In search of multiple domain key recovery. *Journal of Computer Security*, 6:219–235, 1999.
15. Krisis Consortium. The Krisis Project, May 1998.
16. Alfred J. Menezes, Paul C. von Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
17. National Institute of Standards and Technology, Govt. of U.S.A. Requirements for key recovery products. Available at <http://csrc.nist.gov/keyrecovery/>, November 1998. Report of the Technical Advisory Committee to Develop A Federal Information Processing Standard for the Federal Key Management Infrastructure.
18. National Research Council. *Cryptography's Role in Securing the Information Society*. National Academy Press, 1996.
19. Personal communication with the authors.
20. Birgit Pfitzmann and Michael Waidner. How to break fraud-detectable key recovery. *Operating Systems Review, ACM press*, 32(1):23–28, January 1998.
21. U.S. DEPARTMENT OF COMMERCE / National Institute of Standards and Technology. *Federal Information Processing Standard 185—Escrowed Encryption Standard*, February 1994.
22. Eric R. Verheul and Henk C.A. van Tilborg. Binding ElGamal: A fraud-detectable alternative to Key-Escrow proposals. In *Advances in Cryptology - EUROCRYPT'97*, Lecture Notes in Computer Science, pages 119–133. Springer-Verlag 1997, 1997.
23. Adam Young and Moti Yung. Auto-recoverable and auto-certifiable cryptosystems. In Kaisa Nyberg, editor, *Advances in Cryptology, EUROCRYPT'98*, Lecture Notes in Computer Science, pages 17–31. Springer-Verlag, 1998.

A Key Escrow Scheme with Time-Limited Monitoring for One-Way Communication

Masayuki Abe and Masayuki Kanda

NTT Laboratories

1-1 Hikari-no-oka, Yokosuka-shi, 239-0847, Japan

Abstract. This paper studies several of the properties necessary for public key based escrow schemes, and observes that previous schemes lack some important properties. Focusing on the type of communication typified by e-mail, we construct a novel and simple scheme that provides “warrant bounds”, “admissibility”, “surveillance switching”, “non-directive monitoring”, “off-line agency”, “target hiding” and many other useful properties *all at the same time*.

1 Introduction

1.1 Background

With the advance of our computerized society, information security raises many varied demands, some of which can never be fully satisfied simultaneously. Strong ciphers, which protect privacy during communication by rendering tapping useless, have been pursued by many researchers. However, there is strong demand for monitoring communication to combat crime. A common and practical solution for this problem is to use a trusted third party. In a *key escrow scheme*, users have to deposit their private keys with the *escrow agency* (EA in short), which is assumed to disclose the keys to the *law enforcement party* (LEP in short) only if lawfully requested.

One direction of research on key escrow systems is to add useful functionalities. An escrow with *warrant bounds* [6] is a mechanism that limits the period in which the LEP can monitor. *Fraud detectability* [13], or *compliance certification* [5] equivalently, assures the LEP that he and a receiver have the same result of decryption. There are also demands for higher efficiency. The LEP wants the EA to be *off-line* so that they can monitor without the help of the EAs during authorized terms. It may not be acceptable to users to exchange initial messages in one-way communication like e-mail. Several other demands from each of the participants can be listed.

Most of the previous work only concerns some of the important properties listed in this paper. No previous implementation satisfies all of the useful properties *at the same time*. For instance, the scheme in [8] realizes *warrant bounds* and *off-line agency* but does not provide *non-interactive property* (users have to exchange data each other). The scheme in [14] has most of the properties desired

while preserving inter-operability with the existing public key infrastructure. However, it fails to provide off-line agency, which would be strongly demanded by the LEP. As one more example, the scheme in [13] implements *fraud-detectability* but it does not support warrant bounds at all.

1.2 Our Contribution

We first study several properties including the novel notion of *admissibility*, which has been more or less ignored so far despite its importance. We also introduce a novel type of warrant bounds called *threshold warrant bounds* and develop a tool which we call *non-interactive key renewal* to realize the warrant bounds. Then, we construct a key escrow scheme that best suits one type of one-way communication, i.e. e-mail, where most of the properties discussed in section 3 are needed. To be more specific, our scheme provides warrant bounds, admissibility, fraud detectability, sender authentication, shared escrow agency, surveillance switching, non-directive monitoring, off-line agency, non-interactiveness, and target hiding, *all at the same time*, without affecting any of them.

1.3 Organization

Section 2 defines participants and describes our model of the key escrow system. Section 3 discusses the properties and requirements raised in the literature. We then construct a new key escrow scheme in Section 4. The security and efficiency are analyzed in Section 5 and 6. The extensions needed to support shared escrow agencies and the target hiding property are shown in Section 7. A conclusion and some remaining problems are stated in the last section.

2 Model

We assume the following three participants, all of whose computational power is polynomially bounded.

User : An entity who communicates via public channels wherein lawful monitoring is approved. Each session has a sender and a receiver. The two communicating users are assumed not to possess any common secret information beforehand, and also not to communicate via physically private channels. Users may attempt to avoid law enforcement monitoring without leaving any evidence if possible.

Key Escrow Agency (EA): an entity that stores users' private keys. The EA is assumed to never leak any useful information from which the private key of any user could be determined unless a legitimate warrant is shown. We stress that the agency is trusted only in this manner (see discussion on admissibility in Section 3). We will cover the use of shared agencies which cooperatively maintain the private keys of users so that at least a quorum of agencies must commit dishonest behaviour to leak information about the private key. Some of the

previous works, e.g., [7], consider a hierarchical agency structure where each agency handles a different group of users, but we do not assume such participants in this paper.

Law Enforcement Party (LEP): An organization that conducts lawful monitoring in accordance with a legitimate warrant. The LEP may deviate from the protocols in an arbitrary way. Although issuing warrants is out of our scope, we assume that warrants are unforgeable.

Lawful Authority (LA): A party which, after it approves the law enforcement action, issues a warrant if requested. This party is trusted in such a way that it never issues a warrant in conspiracy with a faulty LEP.

Certificate Authority (CA): A party which authorizes the ownership of a public key. This party is a part of the public key infrastructure that maintains and distributes the public keys. This party is trusted in such a way that it never issues a certificate to a fictitious user or a user who does not know the corresponding private key.

Our scenario consists of four individual protocols. First, a user obtains an authorized public key and a corresponding private key. He then registers the key pair to the EA via the *key registration protocol*. When the LEP deems monitoring necessary, it first obtains a warrant from the LA. The LEP then executes the *disclosure protocol* with the EA to get sufficient information (possibly a decryption key) with regard to the target user(s). In the *communication protocol*, a sender encrypts a message and sends it to a receiver who can correctly retrieve the message. While the target user is communicating, the law enforcement party invokes the *monitoring protocol* to obtain the decrypted message.

We assume that communication is performed over a synchronous network. Accordingly, it is assumed that messages are delivered after a delay that is small compared to the term to be explained shortly. For instance, a message that arrives after certain period of time is to be discarded by the receiver.

3 Desirable Properties

This section lists several requirements or preferable properties and analyzes some of the previous works with regard to those properties.

Warrant Bounds : It should be possible to limit lawful activities against the same user within some terms. Otherwise, once a private key is given to the LEP, they can keep monitoring during the lifetime of the private key. This could last up to a few years. Thus, when warrants specify the term of approved monitoring, we need a mechanism that prevents the LEP from performing undue monitoring beyond that term. Some approaches can be seen in [8,2].

Admissibility : Suppose that a faulty message that does not meet expected redundancy is observed during lawful monitoring. Admissibility assures that the redundancy violation has been caused by the sender of the message. This property can be also described as verifiability for the outputs of the LEP and the

EA, and undeniability of the message sender. Namely, to admit the redundancy violation as evidence of faulty behaviour of the sender, one has to assure that the message is sent from the sender, the decryption key is correct, and the decryption procedure is correct. However, the correctness of the decryption key may not be easily assured when some advanced properties are incorporated.

It should be stressed that any entity can cause Byzantine faults because of hardware or human errors, although it may happen only rarely. In particular, systems that deal with highly sensitive matters like key escrow must be designed so that every output is verifiable. One must be aware of the substantial difference between trusting one entity with regard to the confidentiality of stored data and trusting him on the accuracy of his computation. Thus, one can conclude that the EA is trusted too much in some schemes where the outputs are not verifiable. This property is often not considered despite its importance.

Fraud detectability : Attempts at disabling lawful monitoring by letting the LEP recover bogus session keys must be detected, or they should result in unsuccessful communication with an honest receiver. This will be a deterrent to passive deviation by dishonest senders. Such a notion was introduced in [5] and implemented in [13]. We say that *private fraud detectability* is achieved if such fraud is detectable only by entities who have the correct private keys. As an alternative, *public fraud detectability*, where anybody can detect fraud, may be considered. Unfortunately, there is a generic attack by Waidner and Pfitzmann [10] that breaks wide class of public fraud detectable schemes.

Shared escrow agency : The escrow agency can be replaced by multiple agencies so that at least a quorum of agencies must commit dishonest behaviour to leak information about the submitted private keys. Verifiable secret sharing would be a key technique for this property but it may not be sufficient. All computational tasks of the key escrow agency must be easily sharable (and verifiable for admissibility) as well. For instance, it is not known how to efficiently share a hash function such as MD5. In theory, the generic approach of [1] makes it possible, but it turns out to be unacceptably inefficient in practice. Thus, schemes where the escrow agency computes secrets using hash functions do not seem likely to provide this property.

Target hiding: To weaken the trust for the EA and to make the LEP more independent, it is necessary to hide the target of the surveillance from the EA. Some schemes that provide this property employ so-called “blind decoding”, which allows cryptanalysis against the EA’s encryption system. However, known implementations of this property greatly limit other properties such as surveillance switching capability or off-line agency. We will show a plug-in solution that is consistent with other properties.

Surveillance switching: Lawful monitoring may have to be limited to a line between specific users. On the contrary, it may be necessary to monitor all communication that involves a specific user regardless of the partner. We call these two types of surveillance “edge surveillance” and “node surveillance” respectively. A key escrow scheme should seamlessly provide both types of surveillance.

Non-directive monitoring: In the case of node surveillance, monitoring should be possible regardless of the direction of the message flow. This property is unnecessary if the target communication essentially consists of message flow in one direction. For instance, the LEP needs to monitor only outgoing messages from an FTP server to see whether it is exporting banned contents.

Offline EA: Once the LEP obtains a private key, or necessary information for decryption, the LEP should be able to monitor without the help of the EA. Some models, e.g., in [12,11], do not provide this property as the LEP must access the EA for each session.

Sender Authentication : It should be possible for the LEP to identify the sender of a message to prevent a dishonest sender from impersonating another user. This property is also related to fraud detectability and admissibility. If authentication is not provided, no one can be sure whether the sender is honest because he could have been framed. Putting the sender's signature on the encrypted message is sufficient for this purpose.

Noninteractiveness: No interaction should be necessary between communicating users. This property is important for the type of communication where each session contains one message flow from one side to another.

In addition to these properties, some design principles for secure systems must be confirmed. For instance, in [13] users encrypt session keys with the public key of a server. Such systems must be avoided because they totally collapse if the server's public key is compromised.

4 Proposed Scheme

The following setting, based on the discrete logarithm problem, is used in our construction. Let p, q be primes that satisfy $q|p-1$. Let $\mathbb{G}_q$ be a multiplicative subgroup of order q in $\mathbb{Z}_p$. We assume the use of sufficiently large p and q where the Diffie-Hellman problem in $\mathbb{G}_q$ is intractable. By g we denote a generator of $\mathbb{G}_q$. We also assume the use of a one-way hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^{|q|}$ where $|q|$ denotes the length of q in bits. All arithmetic will be done in $\mathbb{Z}_p$ hereafter unless otherwise stated. By $\mathcal{U}$ we denote a set of users.

4.1 Threshold Warrant Bounds

Before describing technical details, we introduce a novel and practical variant of warrant bounds that we call *threshold warrant bounds*. Suppose that the lifetime of a public key is divided into P terms where terms are labelled as $1, \dots, P$. By $\mathcal{P}$ we denote $\{1, \dots, P\}$. A user is assumed to execute the registration protocol within P terms. Warrant bounds can be understood as a property that satisfies the following. Let $\mathcal{T}$ be a set of terms wherein lawful monitoring is approved against a user.

- For any $\mathcal{T} \subseteq \mathcal{P}$, the LEP can monitor in term τ if and only if $\tau \in \mathcal{T}$.

Now we consider the total period of monitoring the same user. If the approved period is short, warrant bounds are important because they prevent privacy violation. If long term surveillance is approved, however, privacy has already been lost. Thus, assuming that long term monitoring is applied only to serious cases and short term monitoring is applied to usual cases, we can establish a sort of threshold that separates these two types of monitoring. Let $t \leq P$ be a constant. Threshold warrant bounds imply the following properties.

- (Short term monitoring) For any $\mathcal{T} \subseteq \mathcal{P}$ such that $|\mathcal{T}| \leq t$, the LEP can monitor in term τ if and only if $\tau \in \mathcal{T}$.
- (Long term monitoring) For any $\mathcal{T} \subseteq \mathcal{P}$ such that $|\mathcal{T}| > t$, the LEP can monitor in arbitrary terms.

For instance, if the lifetime of a public key is a year and each term represents a week, i.e., $P = 52$, and if $t = 4$, monitoring for up to one month (four weeks) a year should be approved for usual cases, and the LEP can not monitor in the remaining weeks. Since approval for more than four weeks is equivalent to the period of a year, it is assumed to be applied only to serious cases. Note that as long as $|\mathcal{T}| \leq t$, the terms can be discrete.

4.2 Noninteractive Key Renewal

Our approach to realize warrant bounds is to have users frequently renew their public and private keys. However, this should be done without interacting with the EA, so that frequent access is avoided. Suppose user u has $t + 1$ key pairs, say $((x_{u0}, y_{u0}), \dots, (x_{ut}, y_{ut}))$ that satisfy $y_{ui} = g^{x_{ui}}$ for $i = 0, \dots, t$. We refer to x_{ui} as the *private base keys* and y_{ui} as the *public base keys* of user u . All the public base keys are certified by the CA and distributed authentically. User u 's public key in term τ is computed as:

$$Y_{u\tau} := \prod_{i=0}^t y_{ui}^{\tau^i}.$$

The corresponding private key is:

$$X_{u\tau} := \sum_{i=0}^t x_{ui} \tau^i \bmod q.$$

That is, $X_{u\tau}$ is the evaluation of a t -degree polynomial $f(Z) := x_{u0} + x_{u1}Z + \dots + x_{ut}Z^t$ in modulo q at evaluation point $Z = \tau$. Note that any entity can compute $Y_{u\tau}$.

In [2] Burmester et al., presented another implementation of noninteractive key renewal based on a novel non-standard assumption. However, unlike ours, only the owner of the public key and the EA who knows the previous private key can produce a new public key while other users can not compute it themselves. Such a property prevents users from “caching” the public keys of other users. Hence access to the CA will increase.

4.3 Main Protocols

Each user $u \in \mathcal{U}$ randomly chooses $t + 1$ private base keys $x_{ui} \in_R \mathbb{Z}_q$ for $i = 0, \dots, t$, and computes the corresponding public base keys $y_{ui} := g^{x_{ui}}$. He then sends all y_{ui} to the CA and proves that he knows the corresponding private base keys by using zero-knowledge interactive proof. Once the procedure is completed correctly, the user's public base key is registered with the public key infrastructure and distributed authentically. A user also registers his public key used for a signature scheme. The corresponding private key must be chosen independently from the private base keys, and it must not be revealed to anybody, including the EA.

When a user registers his/her keys to the EA, the following protocol is performed.

[Registration Protocol]

- R-1.** User u sends (x_{ui}, y_{ui}) for $i = 0, \dots, t$ to the EA.
- R-2.** The EA obtains a certificate from the public key infrastructure and checks whether the public base keys are authorized. It then verifies that $y_{ui} \stackrel{?}{=} g^{x_{ui}}$ for all $i = 0, \dots, t$. If successful, the EA stores everything received.

[End]

To conduct surveillance, the LEP must first obtain a warrant that specifies the target user(s) and term(s). Let τ be the target term wherein monitoring is approved. For node surveillance, the LEP needs $X_{u\tau}$. The following is the disclosure protocol for node surveillance against user u .

[Disclosure Protocol (Node)]

- Dn-1.** The EA computes $X_{u\tau} := \sum_{i=0}^t x_{ui} \tau^i \bmod q$ and sends it to the LEP.
- Dn-2.** The LEP computes $Y_{u\tau} := \prod_{i=0}^t y_{ui}^{\tau^i}$ and verifies that $g^{X_{u\tau}} \stackrel{?}{=} Y_{u\tau}$.

[End]

For edge surveillance between user u and v , the LEP needs the common key $K_{uv\tau}$. However, unlike the disclosure protocol for node surveillance, it is difficult for the LEP to verify the correctness of the obtained $K_{uv\tau}$ without knowing the corresponding $X_{u\tau}$ or $X_{v\tau}$. In the following disclosure protocol for edge surveillance, the EA provides a proof for relation $\log_g Y_{u\tau} = \log_{Y_{v\tau}} K_{uv\tau}$ by using the Chaum-Pedersen protocol [3].

[Disclosure Protocol (Edge)]

- De-1.** The EA computes $X_{u\tau} := \sum_{i=0}^t x_{ui} \tau^i \bmod q$, $Y_{v\tau} := \prod_{i=0}^t y_{vi}^{\tau^i}$ and $K_{uv\tau} := Y_{v\tau}^{X_{u\tau}}$. It chooses $w \in_R \mathbb{Z}_q$ and computes $c := \mathcal{H}(K_{uv\tau} || g^w || Y_{v\tau}^w) \bmod q$ and $z := w - cX_{u\tau} \bmod q$. The EA sends $K_{uv\tau}$ and (c, z) to the LEP.

De-2. The LEP computes $Y_{u\tau} := \prod_{i=0}^t y_{ui}^{\tau^i}$ and $Y_{v\tau} := \prod_{i=0}^t y_{vi}^{\tau^i}$. It then verifies $c \stackrel{?}{=} \mathcal{H}(K_{uv\tau} || g^z Y_{u\tau}^c || Y_{v\tau}^z K_{uv\tau}^c) \pmod{q}$.

[End]

When sender u sends a message to receiver v , they follow the next procedure. Let $\sigma_u(msg)$ be a signature for message msg to be sent (the signing keys must be independent of the deposited base keys). By $E_{key}(msg)$ we denote a symmetrical encryption of msg with common key key . Messages are assumed to have appropriate redundancy.

[Communication Protocol]

C-1. Sender u computes $K_{uv\tau} := Y_{v\tau}^{X_{u\tau}}$.

C-2. Sender u then sends

$$(E_S(M) || E_{K_{uv\tau}}(S) || \sigma_u(E_S(M) || E_{K_{uv\tau}}(S)))$$

to receiver v where M is a message to be received, S is a session key selected randomly, and σ_u is a signature for the previous two parts.

C-3. Receiver v verifies σ_u with u 's public signature verification key. Then, v computes his private key $X_{v\tau}$ and u 's public key $Y_{u\tau}$. He computes $K_{uv\tau} = Y_{u\tau}^{X_{v\tau}}$ and decrypts $E_{K_{uv\tau}}(S)$ with $K_{uv\tau}$ to retrieve session key S . He then decrypts $E_S(M)$ with S to recover M . If S or M does not have proper redundancy, discard everything received.

[End]

Monitoring is done as follows. The LEP first verifies σ with the sender's signature verification key. For node surveillance, the LEP computes $K_{uv\tau} = Y_{v\tau}^{X_{u\tau}}$ and decrypts $E_{K_{uv\tau}}(S)$. For edge surveillance, use $K_{uv\tau}$ given from the EA to decrypt $E_{K_{uv\tau}}(S)$ to get S . Finally, message M is obtained by decrypting $E_S(M)$ with S . If a redundancy violation is observed, store everything and stop (legal action should then be taken).

5 Security Analysis

Here we restrict ourselves to prove security on warrant bounds as it is not clear if warrant bounds is achieved when node and edge surveillance are conducted in a mixed way. (It is not hard to verify that the proposed scheme implements verifiability, surveillance switching, non-directive monitoring, off-line EA, and non-interactiveness.)

Let $\mathcal{N}_u$ denote a list of terms wherein node surveillance was conducted against user u . Similarly let $\mathcal{E}_{uv}$ be a list of terms wherein edge surveillance are performed against user u and v . When an edge surveillance is conducted at term τ , we assume that τ is listed in either $\mathcal{E}_{uv}$ or $\mathcal{E}_{vu}$. Namely, we assume $\mathcal{E}_{uv} \cap \mathcal{E}_{vu} = \emptyset$.

Theorem 1. *If $|\mathcal{N}_u| > t$, the LEP can compute $X_{u\tau}$ for arbitrary $\tau \in \mathcal{P}$.*

Theorem 1 immediately holds from the threshold property of our noninteractive key renewal.

Theorem 2. *If $|\mathcal{N}_u \cup \mathcal{N}_v \cup \mathcal{E}_{uv} \cup \mathcal{E}_{vu}| \geq 2t + 1$, the LEP can compute $K_{uv\tau}$ for arbitrary $\tau \in \mathcal{P}$.*

Proof. Let $f_u(Z)$ and $f_v(Z)$ be random t -degree polynomials decided by the base private keys of user u and v , respectively. Observe that $\log_g K_{uv\tau}$ is on $2t$ -degree polynomial $f_u(Z)f_v(Z)$ at $Z = \tau$. Let W_i be an element of $\mathbb{Z}_p$ such that $\log_g W_i$ is the i -th coefficient of $f_u(Z)f_v(Z)$. Observe that $K_{uvk} = W_0 W_1^k \cdots W_{2t}^{k^{2t}}$ holds. Hence, given K_{uvk} for more than $2t + 1$ different k 's, one can determine all W_i by solving a series of multiplicative equations (without solving the discrete logarithms). Then, one can compute $K_{uv\tau}$ for any $\tau \in \mathcal{P} \subseteq \mathbb{Z}_q$ with those W_i .

Next we state the conditions on $\mathcal{N}$ and $\mathcal{E}$ to force warrant bounds on the LEP. Suppose that the LEP attempts to conduct illegal monitoring at unauthorized terms or against unauthorized users. What the LEP has at this time is the history (keys and proofs) of the previous surveillance. A history, denoted by $Hist$, consists of keys for node investigation, keys and proofs (issued by the EA) for edge investigation, and public base keys for all users. More precisely, $Hist$ is defined as

$$\bigcup_{i,j \in \mathcal{U}} \{ \{X_{ik} \mid k \in \mathcal{N}_i\} \cup \{K_{ijk}, Proof_{ijk} \mid k \in \mathcal{E}_{ij} \cup \mathcal{E}_{ji}\} \cup \{y_{i0}, \dots, y_{it}\} \}$$

where $Proof_{ijk}$ is (c, z) shown in the disclosure protocol for edge surveillance in the previous section. Given such $Hist$, the LEP attempts to compute $X_{u\tau}$ or $K_{uv\tau}$ for some (u, v, τ) that has not appeared in $Hist$. The following theorem states that the LEP will not be successful in such an attempt. In this theorem, we consider the LEP *static* so that it does not *adaptively* construct the history after seeing the base keys of users. Instead, we prove that, *given any history*, the LEP will fail to conduct unauthorized surveillance as long as all $\mathcal{N}_i$'s and $\mathcal{E}_{ij}$'s satisfy the conditions shown below.

Theorem 3. *(Security against unauthorized node surveillance) Let LEP be a probabilistic polynomial time algorithm. Assume that the following conditions are satisfied: for all $i \in \mathcal{U}$,*

$$|\mathcal{N}_i| \leq t, \tag{1}$$

$$|\mathcal{E}_{ij}| \leq t \text{ for all } j \in \mathcal{U} \setminus \{i\}, \text{ and} \tag{2}$$

$$|\mathcal{N}_i \cup \{\bigcup_{j \in \mathcal{U}} \mathcal{E}_{ij}\}| \leq t. \tag{3}$$

Given $\{X_{ik} \mid k \in \mathcal{N}_i\}$, $\{K_{ijk} \mid k \in \mathcal{E}_{ij}\}$ and $\{y_{i0}, \dots, y_{it}\}$ for all $i, j \in \mathcal{U}$, the probability that LEP outputs $X_{u\tau}$ for some $u \in \mathcal{U}$ and $\tau \in \mathcal{P}$ such that $\tau \notin \mathcal{N}_u$ is negligible under the intractability assumption of the discrete logarithm problem.

Proof. Proof is done by contradiction. We construct a discrete logarithm solver M_{DL} assuming the existence of a machine, say M_{node} , that outputs a key for illegal node surveillance. M_{DL} is given an instance (p, q, g, Y) where $\log_g Y$ in Z_p is to be solved. M_{DL} simulates $Hist$ by producing all base keys and works as follows.

Step 1. (Generating the history of targets and periods) M_{DL} randomly generates $\mathcal{N}_i$ and $\mathcal{E}_{ij}$ for all $i, j \in \mathcal{U}$ so that they satisfy the conditions addressed in the statement.

The following steps are repeated for all $i \in \mathcal{U}$.

Step 2. (Generating the base keys) Pick $r_i \in_R Z_q^*$ and compute $y_{i0} := Y^{r_i}$. Next, if $|\mathcal{N}_i \cup \{\bigcup_{j \in \mathcal{U}} \mathcal{E}_{ij}\}| < t$ select $\mathcal{C}_i$ from $\{P+1, \dots, q\}$ so that $|\mathcal{C}_i \cup \mathcal{N}_i \cup \{\bigcup_{j \in \mathcal{U}} \mathcal{E}_{ij}\}| = t$. Let $\mathcal{X}_i$ denote the above set $\mathcal{C}_i \cup \mathcal{N}_i \cup \{\bigcup_{j \in \mathcal{U}} \mathcal{E}_{ij}\}$. Then, for every $k \in \mathcal{X}_i$, select $X_{ik} \in_R Z_q$. Then, compute $y_{i\ell}$ for $\ell = 1, \dots, t$ so that they satisfy $g^{X_{ik}} = y_{i0} \prod_{\ell=1}^t y_{i\ell}^{k^\ell}$ for all $k \in \mathcal{X}_i$ (Note that it does not need solution of the discrete logarithm). Add $y_{i0}, \dots, y_{it}$ to $Hist$.

Step 3. (Generating node surveillance keys) For all $k \in \mathcal{N}_i$, add X_{ik} to $Hist$.

Step 4. (Generating edge surveillance keys and proofs) For all $k \in \bigcup_{j \in \mathcal{U}} \mathcal{E}_{ij}$, compute K_{ijk} as $K_{ijk} = \left(\prod_{\ell=0}^t y_{j\ell}^{k^\ell} \right)^{X_{ik}}$, and add them to $Hist$. Also compute appropriate $Proof_{ijk}$ by using X_{ik} and add it to $Hist$.

The resulted $Hist$ uniformly distributes over the space where the real $Hist$ is taken when the LEP randomly decides the targets and periods of surveillance.

Now M_{DL} gives the produced history to M_{node} with the expectation that it will output unauthorized $X_{u\tau}$. Precisely, M_{DL} works as follows.

Step 5. (Invoking M_{node}) Give $Hist$ produced in step 1-4 to M_{node} , and get an output $(u, \tau, X_{u\tau})$. Abort if $g^{X_{u\tau}} \neq \prod_{i=0}^t y_{ui} \tau^i$ happens.

Step 6. (Computing the discrete logarithm of Y by interpolation) Output $X := r_u^{-1} \sum_{k \in \mathcal{Q}} \lambda_{k, \mathcal{Q}} X_{uk} \bmod q$ where $\lambda_{k, \mathcal{Q}} := \prod_{i \in \mathcal{Q}, i \neq k} \frac{i}{i-k} \bmod q$ and $\mathcal{Q} = \mathcal{X}_u \cup \{\tau\}$.

Observe that $\mathcal{C}_u$ is taken from outside of the valid period in Step 2. This avoids the failure that M_{node} outputs $X_{u\tau}$ for $\tau \in \mathcal{C}_u$ in Step 4. Then, it is easy to see, by inspection, that if M_{DL} completes then X equals $\log_g Y$. The success probability of M_{DL} is the same as that of M_{node} .

Theorem 4. (*Security against unauthorized edge surveillance*) Suppose that the LEP is given $\{X_{ik} \mid k \in \mathcal{N}_i\}$, $\{K_{ijk} \mid k \in \mathcal{E}_{ij}\}$ and $\{y_{i0}, \dots, y_{it}\}$ that satisfy conditions 1, 2, and 3 for all $i, j \in \mathcal{U}$. The probability that the LEP outputs $K_{uv\tau}$ for some $u, v \in \mathcal{U}$ and $\tau \in \mathcal{P}$ such that $\tau \notin \mathcal{E}_{uv} \cup \mathcal{N}_u \cup \mathcal{N}_v$ is negligible under the intractability assumption of the Diffie-Hellman problem.

Proof. Similar to the previous proof, we assume the existence of a machine called M_{edge} which takes $Hist$ as an input and outputs, with non-negligible probability, $(u, v, \tau, K_{uv\tau})$ that is not included in $Hist$. By using M_{edge} as a subroutine we construct a Diffie-Hellman problem solver M_{DH} that takes (p, q, g, Y, Z) as an input and outputs $Y^{\log_g Z}$. Although M_{DH} is more complicated than M_{DL} , the description of M_{DH} is the same as that of M_{DL} in the proof of Theorem 3 except for Steps 2, 5 and 6 as shown in the following. In Step 2, M_{DH} first randomly divides the users into two groups, $\mathcal{U}_1$ and $\mathcal{U}_2$ whose size are the same (if the number of users are odd, we allow difference of one element between $\mathcal{U}_1$ and $\mathcal{U}_2$). It then computes $y_{u0} := Y^{r_u}$ for all users $u \in \mathcal{U}_1$ and $y_{v0} := Z^{r_v}$ for all users $v \in \mathcal{U}_2$ where r_u, r_v are taken randomly from $\mathbb{Z}_q^*$. The rest is the same as original Step 2. In Step 5, M_{DH} receives $(u, v, \tau, K_{uv\tau})$ from M_{edge} .

Before describing Step 6, we need to make some observations. According to Step 2, $y_{u1}, \dots, y_{ut}$ satisfy $g^{X_{uk}} = y_{u0}y_{u1}^k \dots y_{ut}^{k^t}$ for all $k \in \mathcal{X}_u$. From those t equations (recall that $|\mathcal{X}_u| = t$), we can represent y_{ui} as $y_{ui} = g^{\alpha_{ui}}y_{u0}^{\beta_{ui}}$ with some constant α_{ui} and β_{ui} in $\mathbb{Z}_q$. The important fact is that M_{DH} can compute those α 's and β 's from $\mathcal{X}_u$ and X_{uk} chosen in Step 2. Hence, $Y_{u\tau}$ can be represented as

$$\begin{aligned} Y_{u\tau} &= \prod_{i=0}^t y_{ui}^{\tau^i} \\ &= g^{\gamma_{u\tau}} y_{u0}^{\xi_{u\tau}} \end{aligned}$$

with appropriate constant $\gamma_{u\tau}$ and $\xi_{u\tau}$ known for M_{DH} . In the same way, M_{DH} can compute $\gamma_{v\tau}$ and $\xi_{v\tau}$ that satisfies $Y_{v\tau} = g^{\gamma_{v\tau}}y_{v0}^{\xi_{v\tau}}$. Observe that

$$\begin{aligned} \log Y_{u\tau} &= \gamma_{u\tau} + \xi_{u\tau} \log y_{u0} \pmod{q}, \text{ and} \\ \log Y_{v\tau} &= \gamma_{v\tau} + \xi_{v\tau} \log y_{v0} \pmod{q} \end{aligned}$$

holds (logarithms are to base g in above equations and the same applies for the rest of this proof). So, by multiplying above equations, we have

$$\begin{aligned} \log y_{u0} \log y_{v0} &= (\log Y_{u\tau} \log Y_{v\tau} - \gamma_{u\tau} \xi_{v\tau} \log y_{v0} - \gamma_{v\tau} \xi_{u\tau} \log y_{u0} - \gamma_{u\tau} \gamma_{v\tau}) / (\xi_{u\tau} \xi_{v\tau} \\ &\pmod{q}). \end{aligned}$$

Thus,

$$\begin{aligned} y_{u0}^{\log y_{v0}} &= \{Y_{u\tau}^{\log Y_{v\tau}} y_{v0}^{-\gamma_{u\tau} \xi_{v\tau}} y_{u0}^{-\gamma_{v\tau} \xi_{u\tau}} g^{\xi_{u\tau} \xi_{v\tau}}\}^{1/\xi_{u\tau} \xi_{v\tau}} \\ &= \{K_{uv\tau} y_{v0}^{-\gamma_{u\tau} \xi_{v\tau}} y_{u0}^{-\gamma_{v\tau} \xi_{u\tau}} g^{\xi_{u\tau} \xi_{v\tau}}\}^{1/\xi_{u\tau} \xi_{v\tau}}. \end{aligned}$$

In the final step M_{DH} aborts if both u and v are in either $\mathcal{U}_1$ or $\mathcal{U}_2$. Otherwise, it computes $\gamma_{u\tau}$, $\xi_{u\tau}$, $\gamma_{v\tau}$, $\xi_{v\tau}$ and outputs

$$\begin{aligned} \{K_{uv\tau} y_{v0}^{-\gamma_{u\tau} \xi_{v\tau}} y_{u0}^{-\gamma_{v\tau} \xi_{u\tau}} g^{\xi_{u\tau} \xi_{v\tau}}\}^{1/r_u r_v \xi_{u\tau} \xi_{v\tau}} &= y_{u0}^{(\log y_{v0})/r_u r_v} \\ &= (y_{u0}^{1/r_u})^{\log y_{v0}^{1/r_v}} \\ &= Y^{\log Z}. \end{aligned}$$

The description of M_{DH} ends.

The success probability of M_{DH} is worse than that of M_{edge} because M_{DH} fails if M_{edge} outputs u, v both of which belong to either $\mathcal{U}_1$ or $\mathcal{U}_2$. Since the public base keys distribute uniformly over the space Z_p^{t+1} whichever it belongs to, $\mathcal{U}_1$ or $\mathcal{U}_2$, M_{edge} has no information about $\mathcal{U}_1$ and $\mathcal{U}_2$. Hence the probability that u and v belong to different groups is

$$1 - \frac{\binom{|\mathcal{U}_1|}{2} + \binom{|\mathcal{U}_2|}{2}}{\binom{U}{2}}.$$

Since we have $\mathcal{U}_1$ and $\mathcal{U}_2$ as (almost) the same size the probability is larger than $1/2$. Thus, the success probability of M_{DH} is more than half of that of M_{edge} , which is not negligible.

Notice that conditions 1,2,3 are not the reverse of the conditions stated in Theorem 1 and 2. Hence they may not be tight. Indeed, those conditions are needed only for technical reasons (to simulate history of surveillance).

6 Efficiency

The computational overhead of the proposed protocol is small: It only requires private key and public key for each term to be computed from the corresponding base keys. This computational overhead grows in factor $t^2 \log P$. For instance, let us take $P = 52$, and $t = 4$ as before. Let k be a variable that represents terms. As term varies from 1 to 52, k needs 7 bits, the number of multiplications in $\mathbb{Z}_p$ for computing the public key of k -th term is $\sum_{i=0}^t i |k| = 70$. Therefore, even if we do not use an advanced method for calculating vector exponentiation, computation of the public key consumes only 70 modular multiplications in $\mathbb{Z}_p$. This takes about one third of the time needed for the modular exponentiation commonly used in other computations if we take $|q| \approx 160$ as is usual. We expect that such a small overhead is acceptable for many computationally weak user terminals. Necessary storage for base keys is linear in t . Accordingly, the underlying public key infrastructure must support t times more public keys. However, communication complexity, except base key distribution, does not increase at all.

7 Extensions

7.1 Shared Escrow Agency

The shared EAs cooperatively work as a single EA as presented before, so the difference appears only in the registration protocol and the disclosure protocol. We assume that users can access each EA via a private channel during the registration protocol.

In the registration protocol, a user gives a share of his base secret keys to each EA. Each EA then verifies that what is received is a correct share of the base secret keys that correspond to the base public keys. Such a requirement is easily satisfied by using verifiable secret sharing [4] in the registration protocol. The disclosure protocol must be modified so that it is done in a shared manner. Let us omit descriptions of these straightforward modifications.

7.2 Hiding Targets from the Escrow Agency

Here we describe two plug-in schemes for hiding targets from the EA while preserving other properties. Our solution is based on [9]. At the end of term $\tau - 1$, the EA encrypts new private key $X_{u\tau}$ of every user by using El Gamal encryption with its encryption key y (for efficiency, we assume that each term has reasonable span). Since El Gamal encryption is malleable, the LA can *blind* the target ciphertext. The blinded ciphertext is sent to the EA via the LEP, and the LEP finally obtains $X_{u\tau}$ by unblinding the output of the EA. More precise description is as follows (in the following, suffix τ is omitted). The EA prepares a key pair (x, y) such that $x \in \mathbb{Z}_{q-1}$ and $y = h^x \bmod q$ where h is a generator of $\mathbb{Z}_q$. At the end of term $\tau - 1$, the EA encrypts, for all $i \in \mathcal{U}$, X_i as $(M_i, G_i) = (X_i y^{t_i} \bmod q, h^{t_i} \bmod q)$ where $t_i \in_R \mathbb{Z}_{q-1}$. It then publishes the resulted ciphertexts. The following is the procedure of disclosure for node surveillance.

[Blind Disclosure Protocol (Node)]

- Step 1.** The LA selects $w_1, w_2 \in_R \mathbb{Z}_{q-1}$. It then blinds (M_u, G_u) as $(\tilde{M}, \tilde{G}) = (M_u y^{w_1+w_2}, G_u h^{w_1})$. It passes $(\tilde{M}, \tilde{G}, w_2)$ to the LEP together with its authorized signature.
- Step 2.** The LEP sends $(\tilde{M}, \tilde{G})$ and the signature of the LA to the EA.
- Step 3.** The EA computes $\tilde{X} := \tilde{M} / \tilde{G}^x \bmod q$ and sends $\tilde{X}$ back to the LEP.
- Step 4.** The LEP computes $X_u := \tilde{X} / y^{w_2} \bmod q$ and verifies that $g^{X_u} \stackrel{?}{=} Y_u$. If it fails, declare reject.

[End]

It is easy to confirm the correctness of the protocol by checking that

$$\begin{aligned} (\tilde{M} / \tilde{G}^x) / y^{w_2} &= (M_u y^{w_1+w_2}) / (G_u h^{w_1})^x y^{w_2} \\ &= X_u y^{t_u+w_1+w_2} / y^{t_u+w_1+w_2} \\ &= X_u. \end{aligned}$$

Furthermore, $(\tilde{M}, \tilde{G})$ is computationally indistinguishable from any (M_i, G_i) under the intractability assumption of the decision Diffie Hellman problem.

The next is the procedure of disclosure for edge surveillance between user u and v .

[Blind Disclosure Protocol (Edge)]

- Step 1.** The LA selects $w_1, w_2 \in_R \mathbb{Z}_{q-1}$ and $w_3 \in_R \mathbb{Z}_q$. It then blinds (M_u, G_u) and Y_v as $(\tilde{M}, \tilde{G}) = (M_u y^{w_1+w_2}, G_u h^{w_1})$ and $\tilde{Y} = Y_v^{w_3}$. It passes $(\tilde{M}, \tilde{G}, w_2, w_3)$ and $\tilde{Y}$ to the LEP together with its authorized signature.
- Step 2.** The LEP sends $(\tilde{M}, \tilde{G})$ and $\tilde{Y}$ to the EA with the signature of the LA.
- Step 3.** The EA computes $\tilde{K} := \tilde{Y}^{\tilde{M}/\tilde{G}^x} \bmod q$ and sends $\tilde{K}$ back to the LEP. Similar to the original disclosure protocol shown in section 4.3, it then provides a proof (c, z) where $c = \mathcal{H}(\tilde{K}, \tilde{Y}^w, g^w)$ and $z = w - c(\tilde{M}/\tilde{G}^x) \bmod q$ for $w \in_R \mathbb{Z}_{q-1}$.
- Step 4.** The LEP verifies the proof and computes $K_{uv} := \tilde{K}^{1/w_3 y^{w_2}} \bmod q$.
- [End]**

Observe that

$$\begin{aligned}
 \tilde{K}^{1/w_3 y^{w_2}} &= \left(\tilde{Y}^{\tilde{M}/\tilde{G}^x} \right)^{1/w_3 y^{w_2}} \\
 &= \left(\tilde{Y}^{1/w_3} \right)^{M_u y^{w_1+w_2} / (G_u h^{w_1})^x y^{w_2}} \\
 &= Y_v^{X_u y^{t_u+w_1+w_2} / (h^x)^{t_u+w_1} y^{w_2}} \\
 &= Y_v^{X_u}.
 \end{aligned}$$

Thus, the LEP can blindly obtain the common key K_{uv} for edge surveillance.

Finally let us point out that such blind decoding approach will allow the LEP and the LA to launch adaptive chosen ciphertext attacks. Though it is not likely that such attacks will completely break an El Gamal cryptosystem, it could endanger the secrecy of keys whose disclosure is unauthorized.

8 Open Problems

- Our construction of target hiding property could be vulnerable against adaptive chosen message attacks as be former solutions. Find an alternative solution that is immune to such attacks while preserving communication (or even computation) complexity and other properties.
- Treatment of mutually distrusted escrow agencies [9,5] was not covered in this paper. Can such a property be implemented without affecting the other properties listed in this paper?

References

1. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th annual ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

2. M. Burmester, Y. Desmedt, and J. Seberry. Equitable key escrow with limited time span (or how to enforce time expiration cryptographically). In K. Ohta and D. Pei, editors, *Advances in Cryptology — ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 380–391. Springer-Verlag, 1998.
3. D. L. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.
4. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.
5. Y. Frankel and M. Yung. Escrow encryption systems revisited: Attacks, analysis and designs. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 222–235. Springer-Verlag, 1995.
6. N. Jefferies, C. Mitchell, and M. Walker. A proposed architecture for trusted third party services. In *Proceedings of Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 98–104. Springer-Verlag, 1995.
7. I. Kuroda, M. Kanda, S. Miyazaki, and K. Sakurai. Notes on international key escrow system. In *The Proceedings of the 1998 Symposium on Cryptography and Information Security*, number 5.2.C in SCIS98, 1998.
8. A. K. Lenstra, P. Winkler, and Y. Yacobi. A key escrow system with warrant bounds. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 197–207. Springer-Verlag, 1995.
9. S. Micali. Fair public key cryptosystems. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 113–138. Springer-Verlag, 1993.
10. B. Pfitzmann and M. Waidner. How to break fraud-detectable key recovery. Technical Report RZ 2970, IBM, March 1997.
11. K. Sakurai, Y. Yamane, S. Miyazaki, and T. Inoue. A key escrow system with protecting user's privacy by blind decoding. In *1997 Information Security Workshop*, pages 87–96, 1997.
12. Y. Sameshima. A key escrow system of the RSA cryptosystem. In *1997 Information Security Workshop*, pages 75–85, 1997.
13. E. R. Verheul and Henk C. A. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 119–133. Springer-Verlag, 1997.
14. A. Young and M. Yung. Auto-recoverable auto-certifiable cryptosystems. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 17–31. Springer-Verlag, 1998.

Key Management for Secure Multicast with Dynamic Controller

Hartono Kurnio, Rei Safavi-Naini, Willy Susilo, and Huaxiong Wang

Centre for Computer Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, AUSTRALIA
{hk22, rei, wsusilo, huaxiong}@uow.edu.au

Abstract. Multicast security has been intensively studied in recent years. In all previous schemes a trusted group controller is essential. However this is a restrictive assumption in applications such as teleconferences among members of a dynamic group. This leads to the problem of constructing *dynamic* group controllers in the multicast environment. That is, after the initialisation of the system, each user in the group can establish a common key among a subset of users by broadcasting a message (or messages) such that only the users in the designated group are able to decrypt it. In this paper, we present two solutions, one based on *key distribution patterns* which can cater for both private key and public key settings, and the other using *tree based key hierarchy structure* by employing the Diffie-Hellman key exchange protocol. We assess security and efficiency of our proposed schemes.

1 Introduction

Multicast, or *one-to-many* communication is the basic form of transmission in group communication applications and forms the main primitive for a range of advanced telecommunication services including video broadcasting, multi-party teleconferencing, stock quote distribution, and updating software.

Multicast security has been intensively studied in recent years (see, for example [2,3,4,8,10,15,16,17]). Secure communication in multicast environment is much more challenging than traditional point-to-point communication and raises numerous new security problems. Examples are controlling access to the encrypted data, and efficient management of dynamic groups where new members join or existing members need to be evicted.

A simple solution to providing secure communication in a group is by employing conventional point-to-point cryptographic protocols. For example, secure group communication can be achieved by giving each user a pair of public and secret key which can be used to encrypt messages. However this is very inefficient: a user who wants to encrypt a message for the group must encrypt it for each group member individually and then broadcast the concatenation of the encrypted parts. A second solution is to share a common key among the

group members and use the key to perform the cryptographic operation. This raises the question of how to efficiently add new members to, or remove members from, the group such that security of the previous and future communication is guaranteed. When a new user joins the group, the common key can be sent to the new user using secure unicast. However this means that the new user can read all the previous encrypted messages. To keep the previous communications secret from the new user, a new common key can be generated and sent to the old group members encrypted with the old common key, and to the new user using secure unicast. Removing users is a more difficult problem. When users leave the group it is essential to change the group key in order to conceal future communications from the evicted users. This is known as *users revocation* or *blacklisting* problem.

A simple solution to user revocation problem exists when each user in the group shares an individual secret key with a centre which controls the group. When a user is to be deleted from the group, the centre chooses a new common key to be used for encrypting the future group messages, encrypts it with the secret key of each user and sends it to them.

In this system the group controller is the trust and communication bottleneck of the system. The controller knows all the keys used by the group members and its compromise results in the complete loss of system security. It is also communication bottleneck of the system and any user revocation requires its participation.

In this paper we consider revocation problem without the need for a trusted group controller. That is, we allow any member of the group to remove a subgroup of participants and obtain a shared key with the remaining group members. This can be used to establish conferences among arbitrary subgroup, and initiated by a group member.

1.1 Previous Work

The study of encrypted multicast to target some selected subsets of users can be dated to the beginning of 90's by Berkovits [1]. A number of works have followed, focusing mainly on the problem of security multicast for the private-key case, under the assumption that one-way functions exist.

A major step in this direction has been the introduction of *broadcast encryption* system [8] for pay-TV application. The system enables a single (fixed) sender to transmit a message to a dynamically changing subgroup of users, such that collusion of at most t users cannot decrypt the transmissions, unless one of the collusers is a member of the subset. The group may change and this change will be affected through a new *control block*. A very important feature of this scheme is that the overhead of the control block does not depend on the number of the users that are removed from the group. The communication overhead of the scheme, assuming the existence of one way function, is $O(t^2 \log_2 N \log_2^2 t)$, and each user needs to maintain $O(t(\log_2 N)^2 \log_2 t)$ keys, where N is the total number of users in the group. A number of papers studied various variants

of broadcast encryption and user revocation problem with no computational assumptions made, see for example, [15,10,3].

A totally different solution was proposed by Wallner, Harder and Agee [16], and independently by Wong, Gouda and Lam [17]. They propose a tree-based scheme that allows removal of a single user with $\log_2 N$ re-keying messages. Re-keying message can be repeated many times for removal of multiple users. The scheme provides protection against collusion of arbitrary number of users. To exclude t users, the scheme requires $O(N)$ keys for the group controller, $O(N)$ transmission overhead and requires only $O(\log_2 N)$ keys for each user. Canetti et al [2] show how to use pseudo-random generators to reduce the transmission overhead by a constant factor, but still require $O(N)$ keys for the group controller. Canetti, Malkin and Nissim [3] explore possible trade-off between the transmission overhead, user keys and centre keys. More recently, Chang et al extended the scheme of [16], using Boolean function minimisation techniques [4]. They reduced the total number of centre keys from $O(N)$ of Wallner et al's scheme to $O(\log_2 N)$, while retaining the same order transmission overhead and user keys. However, the main drawback of Chang et al scheme is that it does not protect against collusion attack: that is collusion of two users can reveal all the keys of the system.

2 Our Work

All previous schemes have a single *group controller* who is trusted. The group controller generates and distributes users' keys in the system setup, and manages the key updates. In many applications, such as dynamic conferences, group users may wish to transmit data to a subgroup of users. If the single group controller model is used, then all communication from users should be sent through a unicast channel to the group controller and from there multicasted to the group. This raises numerous problems such as single point of failure, communication overhead for the group controller, and communication delay.

In contrast to the previous solution in which the controller is fixed, in this work we consider the scenario that the group controller is *dynamic*. That is, after the initialisation of the system, each user can establish a common key for a subset of the group. This is achieved by a single broadcast to the group which can only be decrypted by the members of the target subgroup.

A simple solution to the problem of dynamic controller is to employ the scheme of a single controller as a building block and associate a single controller scheme to each user such that the rest of the group are the receivers. We call this simple solution the *trivial scheme*. The obvious drawback of this solution is that the key storage of each user is prohibitively large, that is $N - 1$ times the storage of a receiver plus the storage of a controller in a single controller scheme, which is linear in N , and for large groups is very inefficient.

We address the problem by presenting two approaches, one based on key distribution patterns which can cater for both private-key and public-key settings,

and the other one based on key hierarchy by employing the Diffie-Hellman key exchange protocol.

The basic idea of the first approach is to design the structure of keys in such a way that each user holds a different subset of keys, such that each pair of users share at least a key that is not known to any removed users – a similar method has been used in the fixed sender scheme [10]. In this basic scheme, for a group of N users the number of keys to be stored by each user is $O(\log_2 N)$ and the communication overhead for deleting t users is only $(t \log_2 N)$. The scheme provides security against collusion of up to t users.

The second approach is inspired by the work of Wallner et al [16]. We use a logical tree structure with each leaf corresponding to a single user. Nodes in the tree correspond to intermediate keys and the root is the key known by all the users. However, unlike Wallner et al, we do not require a group controller neither for key generation and distribution in the system setup, nor user revocation. For key allocation, we apply binary key structure. Every user holds a different subset of keys and maintains the keys and the tree structure for key update. Each user has storage complexity $O(\log_2 N)$, and communication complexity for removing t users ranges from 0 to $O(N)$ depending on the relative positions of users in the logical tree. This approach is also secure against collusion of any t users, where t can be any number less than N .

2.1 Notations

The ring of integers modulo a number n is denoted by Z_n , and the multiplicative subgroup of integers relatively prime to n , by Z_n^* .

3 The First Approach

Assume there are N users, $\mathcal{U} = \{\mathcal{U}_1, \dots, \mathcal{U}_N\}$. There is a Trusted Authority (TA) who distributes keys to each user during the system setup. At a later time the users in the group can broadcast messages such that only some designated users can decrypt the messages.

3.1 The Scheme

The notion of a *key distribution pattern* is central to the first approach.

Definition 1. Let $X = \{x_1, \dots, x_n\}$ and $\mathcal{B} = \{B_1, \dots, B_N\}$ be a family of subsets of X . The pair $(X, \mathcal{B})$ is called an (n, N, t) -key distribution pattern $((n, N, t)$ -KDP) if

$$|(B_i \cap B_j) \setminus (\cup_{k=1}^t B_{s_k})| \geq 1$$

for any $(t+2)$ -subset $\{i, j, s_1, \dots, s_t\}$ of $\{1, 2, \dots, N\}$.

Key distribution patterns (KDP) [11] are finite incidence structures that were originally designed to distribute keys between pairs of participants in a network and in the absence of an online key distribution centre. A KDP is used to allocate a collection of subkeys to users in a system such that any pair of users can compute a common key by finding an appropriate combination of their subkeys.

In our first scheme we use an (n, N, t) -KDP and allow users to determine a common key for every $N - t$ users. Let $(X, \mathcal{B})$ be an (n, N, t) -KDP. In the system setup, the TA randomly selects a set of n keys $k_1, \dots, k_n$ and for each user $\mathcal{U}_i$ gives him a subset $\mathcal{K}_i = \{k_r \mid x_r \in B_i\}$ of keys. Assume that a user $\mathcal{U}_i$ wants to establish a session key SK with other users of the group except t users, say $\mathcal{U}_{\ell_1}, \dots, \mathcal{U}_{\ell_t}$. The user $\mathcal{U}_i$ encrypts the session key SK with all his keys except those keys incident to $\mathcal{U}_{\ell_1}, \dots, \mathcal{U}_{\ell_t}$, and broadcasts the encrypted message. That is, $\mathcal{U}_i$ broadcasts $\{E_{k_r}(SK) \mid r \in B_i \setminus (B_{\ell_1} \cup \dots \cup B_{\ell_t})\}$. From the definition of KDP we know that every $\mathcal{U}_j \in \mathcal{U} \setminus \{\mathcal{U}_{\ell_1}, \dots, \mathcal{U}_{\ell_t}\}$ has at least one key k_r where $x_r \in (B_i \setminus (B_{\ell_1} \cup \dots \cup B_{\ell_t}))$, and so he can decrypt $E_{k_r}(SK)$ to obtain SK , while every $\mathcal{U}_{\ell_j}, 1 \leq j \leq t$ can not decrypt the message since he does not have any of the keys used for encryption.

Theorem 1. *The above scheme based on an (n, N, t) -KDP can remove up to t users from a group of N users. It requires each user to store less than n keys, and the maximum number of transmissions is n . The system is secure against collusion of t malicious users.*

The theorem shows that the overhead of the system is affected by various parameters of the underlying key distribution pattern. For given t and N , we expect n to be as small as possible. Equivalently, for given n and t , we expect N to be as large as possible.

Constructing KDP with maximal N for a given size n has been extensively studied in literature; see [11, 9, 13, 6, 12, 15]. Mitchell and Piper [11], and Gong and Wheeler [9] gave explicit constructions for (n, N, t) -KDP in which n is $O(N)$, as opposed to the trivial construction that gives each pair of users an individual key, which requires that n is $O(N^2)$. Dyer, Fenner, Frieze and Thomason [6] showed the existence of (n, N, t) -KDP with $n = O(\log_2 N)$.

The above analysis shows that the number of keys that needs to be stored by each user is of the order $O(\log_2 N)$ and that the length of the message needed for updating the session key and removing t users is $O(\log_2 N)$, where the message is concatenation of the session key encrypted with a number of keys. The following example gives an illustration of the above re-keying scheme.

Example 1.

Let $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $\mathcal{B} = \{B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}, B_{11}, B_{12}\}$ defined as follows.

$$\begin{aligned} B_1 &= \{4, 5, 6, 7, 8, 9\} & B_2 &= \{2, 3, 5, 6, 8, 9\} \\ B_3 &= \{2, 3, 4, 6, 7, 8\} & B_4 &= \{2, 3, 4, 5, 7, 9\} \\ B_5 &= \{1, 2, 3, 7, 8, 9\} & B_6 &= \{1, 3, 4, 6, 7, 9\} \\ B_7 &= \{1, 3, 4, 5, 8, 9\} & B_8 &= \{1, 3, 5, 6, 7, 8\} \\ B_9 &= \{1, 2, 3, 4, 5, 6\} & B_{10} &= \{1, 2, 4, 5, 7, 8\} \\ B_{11} &= \{1, 2, 5, 6, 7, 9\} & B_{12} &= \{1, 2, 4, 6, 8, 9\}. \end{aligned}$$

Then $(X, \mathcal{B})$ is a $(9, 12, 1)$ -KDP.

Assume that after initialisation, user $\mathcal{U}_1$ wants to establish a common key with the rest of users except $\mathcal{U}_3$. Since $B_1 \setminus B_3 = \{5, 9\}$, using our first scheme, $\mathcal{U}_1$ encrypts the key SK using k_5 and k_9 , and broadcasts $\{E_{k_5}(SK), E_{k_9}(SK)\}$ to the group. It is easy to see that every user except $\mathcal{U}_3$ can decrypt at least one encrypted SK in the broadcasted message using k_5 or k_9 . On the other hand, $\mathcal{U}_3$ does not know k_5 and k_9 and so can not decrypt the message.

3.2 Communication Efficiency

In the following we show how to improve the communication cost of the basic scheme. That is, reduce the bandwidth used by the group controller for broadcasting information.

Our construction uses *erasure codes* – a similar method has been proposed for the single controller case in [10]. Erasure codes are a special class of error-correcting codes that allow recovery of a message if part of it is damaged or erased during the transmission.

Definition 2. An $[n, k, m]_q$ (constructive) erasure code is a polynomial-time function $C : \mathcal{F}_q^k \rightarrow \mathcal{F}_q^n$ such that there exists a polynomial time function $D : \hat{\mathcal{F}}_q^n \rightarrow \mathcal{F}_q^k$, where $\hat{\mathcal{F}}_q = \mathcal{F}_q \cup \{\perp\}$, such that for all $v \in \mathcal{F}_q^k$, if $u \in \hat{\mathcal{F}}_q^n$ is such that u agrees with $C(v)$ on at least m places, and is $\perp$ elsewhere, then $D(u) = v$.

Given an $[n, k, m]_q$ erasure code, one can encode a message v to obtain a codeword $C(v)$. The message v can be reconstructed if up to $n - m$ positions of $C(v)$ are damaged or erased. Erasure codes can be constructed using error-correcting codes, such as Reed-Solomon codes. Given a message vector $v = (v_0, v_1, \dots, v_{k-1}) \in \mathcal{F}_q^k$, we construct the polynomial $p(x) = v_0 + v_1x + \dots + v_{k-1}x^{k-1}$. Let $e_1, e_2, \dots, e_n$ be n distinct elements in $\mathcal{F}_q$. The encoding is defined by $C(v) = (p_v(e_1), p_v(e_2), \dots, p_v(e_n))$, and the decoding D uses k pairs $(e_i, p_v(e_i))$ to interpolate the polynomial and reconstruct the coefficients of $p_v(x)$ and obtain the source message v .

In order to apply the erasure codes to improve the communication cost in our basic scheme, we need to slightly modify the definition of key distribution patterns.

Definition 3. Let $X = \{x_1, \dots, x_n\}$ be a set and $\mathcal{B} = \{B_1, \dots, B_N\}$ be a family of subsets of X . The pair $(X, \mathcal{B})$ is called an (n, N, t) - α -key distribution pattern $((n, N, t)$ - α -KDP) if

$$|(B_i \cap B_j) \setminus (\cup_{k=1}^t B_{s_k})| \geq \alpha$$

for any $(t+2)$ -subset $\{i, j, s_1, \dots, s_t\}$ of $\{1, 2, \dots, N\}$.

For $\alpha = 1$, an $(n, N, t) - \alpha$ -KDP is the same as an (n, N, t) -KDP. It is easy to see that α -KDP with $\alpha \geq 2$ can be constructed by concatenating of multiple KDPs. Also an $(n, N, t) - \alpha$ -KDP is an $(n, N, t-1) - \alpha'$ -KDP with $\alpha' \geq \alpha + 1$. It would be interesting to look for more sophisticated solutions. We believe that most techniques for constructing KDPs can be generalised to α -KDP in a straightforward manner.

Now the basic scheme can be modified as follows. Assume that the (auxiliary) keys of the users are elements of a finite field $\mathcal{F}_q$. Let $(X, \mathcal{B})$ be an (n, N, t) - α -KDP. For simplicity, we assume that $|B_i| = \ell$, for all i , and each session key SK is an element in $\mathcal{F}_q^w$. To update a session key in the basic scheme, user $\mathcal{U}_i$ applies a $[\ell, w, \alpha]_q$ erasure code (C, D) in the generation of the broadcast message. For a session key $SK = (SK_1, SK_2, \dots, SK_w)$, $\mathcal{U}_i$ computes $C(SK) = (c_1, c_2, \dots, c_\ell)$ and broadcasts $E(SK) = \{E_{k_r}(c_r) \mid \text{to all } r \text{ such that } x_r \in B_i \setminus (B_{i_1} \cup B_{i_2} \dots \cup B_{i_t})\}$. Since each non-excluded user has at least α keys from the keys of $\mathcal{U}_i$, it follows that each non-excluded user can decrypt α messages of $E(SK)$ and so can decrypt α components of $C(SK)$, and obtain EK .

In an $[n, k, m]_q$ erasure code, the length of the codeword $C(v)$ is $n \log_2 q$ bits, whereas the length of the source message v is $k \log_2 q$ bits, and hence the rate which indicates the extra bandwidth, is n/k . We note that the basic scheme uses an $[n, 1, 1]_q$ erasure code in the construction. In general, we expect k to be as large as possible to minimise the extra bandwidth. We also note that to use this construction with $k > 1$, the parameter α in the $(n, N, t) - \alpha$ -KDP must satisfy $\alpha \geq k$ which requires larger n for the same values of N and t , and so more keys. A more detailed analysis for this tradeoff is an interesting problem remained open.

4 The Second Approach

In this section, we present a different solution to user revocation problem. In this approach a dynamic conference, based on a binary tree key structure and without the assistance of a trusted centre in the setup phase, is constructed. The conference is dynamic in the sense that users in the group are able to form subgroups and determine a unique session key. Later they can use the session key to have secure communication among themselves. The session key of a subgroup $\mathcal{S}$ is obtained by blacklisting, or removing, $\mathcal{U} \setminus \mathcal{S}$ users. Our work is influenced by [16, 17, 7], and [3]. Although these works also use tree structure, they require a group controller for both the initial setup and membership update.

In order to form a subgroup $\mathcal{S} \subset \mathcal{U}$, users in $\mathcal{S}$ need to form the session key. This means that each user in $\mathcal{S}$ should be able to act as a group controller.

4.1 Basic Construction

The basic idea of this construction is to apply a two-party key exchange protocol to a binary tree structure, and use multicasting as the mode of transmission. Nodes and root of the tree correspond to keys. Each leaf is a user's secret key.

Assume a group $\mathcal{U}$ of N users want to establish a dynamic conference. In the system setup, they agree on the values of parameters to be employed by the two-party key exchange protocol. Each user $\mathcal{U}_i$, $1 \leq i \leq N$, randomly chooses his secret key. They form a binary tree with N leaves and agree on their position on the tree leaves. The tree structure is kept by all users. Note that the tree might be unbalanced. Alternatively, values of parameters and tree structure can be generated by a Trusted Authority (TA) and securely sent to all the users.

In a two-party key agreement protocol, each party has a key k , which is kept secret. User $\mathcal{U}_1$ sends $k'_1 = f_1(k_1)$ to user $\mathcal{U}_2$, and user $\mathcal{U}_2$ sends $k'_2 = f_1(k_2)$ in return. Now they can calculate $k_{1,2} = f_2(k_1, k'_2)$ and $k_{2,1} = f_2(k_2, k'_1)$, respectively. If f_1 and f_2 are chosen such that $k_{1,2} = k_{2,1}$, then the two users will share a key. For a key k , $k' = f_1(k)$ is called the *shadow key* of k .

The node keys in the tree are established in the leaf-to-root direction as follows. Firstly, each user uses his secret key k_1 and the shadow key of his sibling, that is $k'_2 = f_1(k_2)$, to form his parent node key, $k_{1,2} = f_2(k_1, k'_2)$. Then, each user uses his node key, which is $k_{1,2}$, and the shadow of the sibling to his node key, that is $k'_{3,4} = f_1(k_{3,4})$, to form the next level parent node key, $k_{1,2,3,4} = f_2(k_{1,2}, k'_{3,4})$. This process continues for every node key until reaching the root key. The root key is known by all users and is the group key. The shadow of a secret key is multicasted once by its owner. The shadow of a node key, that is known by more than one user, is multicasted once by one of the users that can calculate that key. Each user receives and stores shadows of sibling of his keys. These principles imply that each user stores shadows of the sibling of his node keys (secret key) along his path to the root. As a consequence, each user is able to compute node keys along his path to the root.

For one or multiple removal, a subgroup $\mathcal{S}$ updates their tree structure and keys. The updating is as follows. Leaf of every deleted user is pruned and his parent is deleted. His sibling replaces position of his parent. If a node key has a single child, it is deleted and replaced by its child. If a node key does not have any children, then it is deleted. This guarantees that after the updating process, every secret key and node key has a sibling. To obtain the subgroup key, node keys and the root key, that are known (compromised) by the deleted user(s), are updated. The process is by applying a two-party key exchange protocol to the compromised keys in the leaf-to-root direction. Only shadows of the updated node keys have to be multicasted. Finally, it is straightforward to update the root key. This root key is the subgroup key for $\mathcal{S}$.

For implementing our scheme in this paper, we apply Diffie-Hellman key exchange protocol [5].

4.2 System Setup

Suppose a group $\mathcal{U}$ of N users want to establish a dynamic conference. Firstly, they agree on a prime p and a generator $\alpha \in Z_p^*$. Each user $\mathcal{U}_i$, $1 \leq i \leq N$, randomly chooses an integer $k_i \in Z_p^*$ and keeps it as his secret key. Next, they form a binary tree with the number of leaves equal to N and agree on their position on the leaves. The tree structure is known to all users. One possible structure for the binary tree is shown in Figure 1.

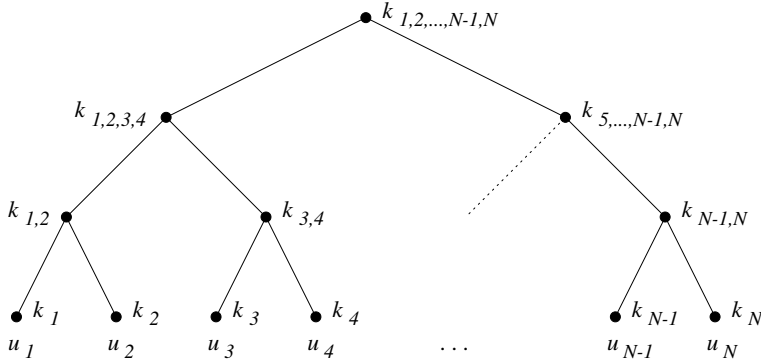


Fig. 1. Key tree structure of $\mathcal{U}$.

Formation of keys in the tree is started from $k_{1,2}, k_{3,4}, \dots$, or $k_{N-1,N}$. User $\mathcal{U}_1$ multicasts $k'_1 = \alpha^{k_1} \pmod{p}$ and $\mathcal{U}_2$ multicasts $k'_2 = \alpha^{k_2} \pmod{p}$. They compute $k_{1,2} = \alpha^{k_1 k_2} \pmod{p}$. Users $\mathcal{U}_3$ and $\mathcal{U}_4$ multicast shadows of their keys and compute $k_{3,4} = \alpha^{k_3 k_4} \pmod{p}$. The rest of the users do the same and finally, users $\mathcal{U}_{N-1}$ and $\mathcal{U}_N$ multicast shadows of their keys and compute $k_{N-1,N} = \alpha^{k_{N-1} k_N} \pmod{p}$.

The process goes up one level. User $\mathcal{U}_1$, or $\mathcal{U}_2$, multicasts $k'_{1,2} = \alpha^{k_{1,2}} \pmod{p}$, and user $\mathcal{U}_3$, or $\mathcal{U}_4$, multicasts $k'_{3,4} = \alpha^{k_{3,4}} \pmod{p}$. Users $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3$, and $\mathcal{U}_4$ compute $k_{1,2,3,4} = \alpha^{k_{1,2} k_{3,4}} \pmod{p} = \alpha^{\alpha^{k_1 k_2} \alpha^{k_3 k_4}} \pmod{p}$. Node key $k_{5,...,N-1,N}$ is formed in the same way by users $\mathcal{U}_5, \dots, \mathcal{U}_{N-1}, \mathcal{U}_N$.

Finally, user $\mathcal{U}_1, \dots$, or $\mathcal{U}_4$ and user $\mathcal{U}_5, \dots$, or $\mathcal{U}_N$ multicast their shadow keys and compute $k_{1,2,...,N-1,N} = \alpha^{k_{1,2,3,4} k_{5,...,N-1,N}} \pmod{p}$ which is the group key for $\mathcal{U}$.

Each user stores shadows of the sibling of his node keys along his path to the root. For example, user $\mathcal{U}_1$ stores $k'_2, k'_{3,4}$, and $k'_{5,...,N-1,N}$ and is able to compute $k_{1,2}, k_{1,2,3,4}$, and $k_{1,2,...,N-1,N}$.

4.3 Removal Process

The key tree of $\mathcal{U}$ consists of the key trees of subgroups of $\mathcal{U}$. Let $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_{N-2}\}$ be the collection of subgroups of $\mathcal{U}$ where $\mathcal{G}_j$ has a session key which is a

node key. The number of such subgroups is equal to the number of nodes in the tree minus root, which is $N - 2$ [14]. For example, subgroup $\mathcal{U}_1, \dots, \mathcal{U}_4$ has the session key $k_{1,2,3,4}$. Users of a $\mathcal{G}_j$ can use the subgroup key to establish secure communication among themselves. However, the number of these subgroups are less than the number of all possible subgroups of $\mathcal{U}$, which is $N - 2 < \sum_{r=2}^N \binom{N}{r}$. When users of a subgroup $\mathcal{S} \neq \mathcal{G}_j$, $1 \leq j \leq N - 2$, want to obtain a subgroup key, they need to remove $\mathcal{U} \setminus \mathcal{S}$ users.

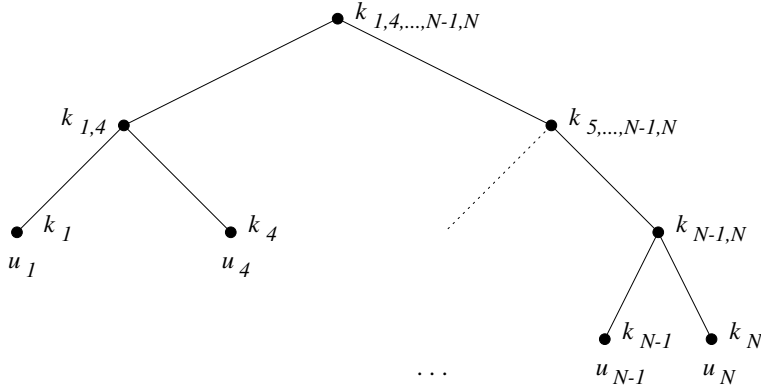


Fig. 2. Key tree structure after removal of $\mathcal{U}_2$ and $\mathcal{U}_3$.

Suppose a subset $\mathcal{S} \subset \mathcal{U}$, $\mathcal{S} = \{\mathcal{U}_1, \mathcal{U}_4, \dots, \mathcal{U}_N\}$ want to establish a subgroup. They delete $\mathcal{U}_2$ and $\mathcal{U}_3$ and update the tree structure to Figure 2. Leaves $\mathcal{U}_2$ and $\mathcal{U}_3$ have been pruned and leaves $\mathcal{U}_1$ and $\mathcal{U}_4$ replace their parents. In the process to obtain session key of the subgroup, keys that have been known by $\mathcal{U}_2$ and $\mathcal{U}_3$ will be updated. In this case, $k_{1,2,3,4}$ and $k_{1,2,\dots,N-1,N}$ in Figure 1 are changed to $k_{1,4}$ and $k_{1,4,\dots,N-1,N}$ in Figure 2, respectively.

In the updating process, user $\mathcal{U}_1$ multicasts $k'_1 = \alpha^{k_1} \pmod{p}$ and $\mathcal{U}_4$ multicasts $k'_4 = \alpha^{k_4} \pmod{p}$. They compute $k_{1,4} = \alpha^{k_1 k_4} \pmod{p}$. User $\mathcal{U}_1$, or $\mathcal{U}_4$, multicasts $k'_{1,4} = \alpha^{k_{1,4}} \pmod{p}$, and $\mathcal{U}_5, \dots, \mathcal{U}_{N-1}, \mathcal{U}_N$ update their keys. Note that user $\mathcal{U}_5, \dots$, or $\mathcal{U}_N$ does not need to multicast $k'_{5,\dots,N-1,N} = \alpha^{k_{5,\dots,N-1,N}} \pmod{p}$ since $k'_{5,\dots,N-1,N}$ is stored by users $\mathcal{U}_1$ and $\mathcal{U}_4$. Finally, $\mathcal{U}_i$, $i = 1, 4, \dots, N - 1, N$, computes $k_{1,4,\dots,N-1,N} = \alpha^{k_{1,4} k_{5,\dots,N-1,N}} \pmod{p}$. This key is the subgroup key for $\mathcal{S}$.

4.4 Efficiency and Security

To evaluate efficiency of the proposed scheme, we consider the following measures: (i) the amount of users' storage, and (ii) the communication cost for determining the session key. Since efficiency is also influenced by the tree structure, for simplicity we assume that the tree structure is binary and balanced.

Theorem 2. *The above scheme based on Diffie-Hellman key exchange protocol can remove any number of users from a group of N users. Each user has to store $O(\log_2 N)$ keys and will have the maximum transmission of $\log_2 N$, while the upper bound on the total number of transmissions in the removal process is $O(N)$. The system is secure against collusion of the removed users provided that Diffie-Hellman problem is hard.*

Proof. Each user needs to store the shadows of the sibling of node keys along his path to the root, which is $\log_2 N$ since we assume that the tree is binary and balanced. We know that in the removal process the shadow of a node key might need to be multicasted to the children of the sibling of that node. If such multicasting is performed by a user with respect to each node along his path to the root, then that user needs to make $\log_2 N$ multicast transmissions. On the other hand, the total number of leaves and nodes in a binary tree, except the root node, is

$$\sum_{z=1}^{\log_2 N} \frac{N}{2^{z-1}} = N \sum_{z=1}^{\log_2 N} (1/2)^{z-1} = N \left(\frac{1 - (1/2)^{\log_2 N}}{1 - 1/2} \right) = 2N - 2.$$

That means that the total number of transmissions in the removal process is bounded by $2N - 2$.

Next, we show that it is computationally infeasible for collusion of all the removed users to find updated node keys and session key. From Diffie-Hellman key exchange protocol, we know that knowing the shadow of a key, it is infeasible to find the key itself. So any subgroup of users, even if they collude, only know the node keys along their paths to the root. Assume $\mathcal{S}$ is the set of users to be removed from the group. During the removal process all the node keys and the session key that were known to the members of $\mathcal{S}$, are updated. Although the removed users know the shadows of some keys from $\mathcal{U} \setminus \mathcal{S}$, the difficulty of Diffie-Hellman problem guarantees that this information will not help them in finding the new session key.

The structure of the tree shows that the root key (i.e. session key) is known to all users, and each node key is known by two or more users. This is true because the shadow of a node key is multicasted once by one user whose path passes through that node. There are $2N - 2$ shadows to be multicasted which can be shared by all N users. Thus we have the following corollary.

Corollary 1. *The average number of transmission for each user is $\lceil (2N - 2)/N \rceil = 2$.*

5 Performance Comparison

In this section we summarise and compare performance of our proposed schemes to that of the trivial scheme. Recall that the trivial scheme is constructed from multiple copies of a single controller scheme (section 2). In this comparison, we

construct the trivial scheme based on the schemes of Fiat-Naor [8], Wallner et al [16], and Chang et al [4]. For efficiency comparison, we focus on performance in terms of the proportions of bandwidth, round, and storage required to update the session key. Bandwidth is measured by the total number of messages where one message contains only one key. The messages might be sent in one round or multiple rounds. Storage is in term of the number of keys a user has to keep.

For security comparison, we concentrate on the size of removal and rounds of removal. Size of removal is the number of users that is removed in one key update such that collusion of the removed users cannot break the security of system. Rounds of removal means the number of times the key update procedure can be employed while the system remains secure.

Table 1. Performance Comparison

	Proposed Schemes		Trivial Scheme, <i>based on</i>		
	First Approach	Second Approach	Fiat-Naor [8]	Wallner et al [16]	Chang et al [4]
Bandwidth	$O(\log N)$	L:0, U: $O(N)$	$O(\log N)$	L:0, U: $O(N)$	L:0, U: $O(N)$
Round	1	L:0, U: $\log N$	1	L:0, U: $\log N$	L:0, U: $\log N$
Storage	$O(\log N)$	$O(\log N)$	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
Size of removal	up to t	up to N	up to t	up to N	1
Rounds of removal	Single	Multiple	Multiple	Multiple	Multiple

Note: L = Lower Bound; U = Upper Bound

Bandwidth usage of the trivial scheme can be obtained from that of the single controller scheme. Both schemes of Wallner et al and Chang et al have lower bound 0 and upper bound $O(N)$ for bandwidth complexity for removing users. This can be proved similar to Theorem 2 and in section 4.3. Round complexity of the trivial scheme follows from that of the single controller scheme. User storage of the trivial scheme equals to $N - 1$ times storage of a leaf user, plus the storage of the controller in single controller scheme. Therefore, user storage of the trivial scheme constructed from Wallner et al scheme is $2N + (N - 1) \log(N - 1) \approx O(N \log N)$ and that of Chang et al is $2 \log N + (N - 1) \log(N - 1) \approx O(N \log N)$. Size and rounds of removal of the trivial scheme follows that of the single controller scheme. Hence, the obvious advantage of our approaches is the reduced user storage.

6 Conclusion

We considered the problem of constructing a *dynamic* group controller in a multicast environment. As we pointed out in section 2, this construction can be achieved by employing a trivial scheme. However, the resulting solution will be inefficient, in terms of key storage. We proposed two new approaches, one based on key distribution patterns, and the other based on key hierarchy structure and showed that they achieve better efficiency compared to the trivial one. We also evaluated the efficiency and security of our proposed schemes.

References

1. S. Berkovits. How to Broadcast a Secret. *Advances in Cryptology – EUROCRYPT '91, Lecture Notes in Computer Science 547*, pages 535-541, 1991.
2. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas. Multicast Security: A Taxonomy and Some Efficient Constructions. *INFOCOM '99*, pages 708-716, 1999.
3. R. Canetti, T. Malkin and K. Nissim. Efficient Communication-Storage Tradeoffs for Multicast Encryption. *Advances in Cryptology – EUROCRYPT '99, Lecture Notes in Computer Science 1592*, pages 459-474, 1999.
4. I. Chang, R. Engel, D. Kandlur, D. Pendarakis and D. Saha. Key Management for Secure Internet Multicast Using Boolean Function Minimisation Techniques. *INFOCOM '99*, pages 689-698, 1999.
5. W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. Inform. Theory*, pages 644-654, vol. 22, 1976.
6. M. Dyer, T. Fenner, A. Frieze and A. Thomason. On Key Storage in Secure Networks. *Journal of Cryptology* **8**, pages 189-200, 1995.
7. D. A. McGrew and A. T. Sherman. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. *Manuscript*, 1998.
8. A. Fiat and M. Naor. Broadcast Encryption. *Advances in Cryptology – CRYPTO '93, Lecture Notes in Computer Science 773*, pages 480-491, 1994.
9. L. Gong and D. J. Wheeler. A Matrix Key-Distribution Scheme. *Journal of Cryptology* **2**, pages 51-59, 1990.
10. R. Kumar, S. Rajagopalan and A. Sahai. Coding Constructions for Blacklisting Problems Without Computational Assumptions. *Advances in Cryptology – CRYPTO '99, Lecture Notes in Computer Science 1666*, pages 609-623, 1999.
11. C. J. Mitchell and F. C. Piper. Key Storage in Secure Networks. *Discrete Applied Mathematics* **21**, pages 215-228, 1988.
12. C. M. O'Keefe. Key Distribution Patterns Using Minkowski Planes. *Designs, Codes and Cryptography* **5**, pages 261-267, 1995.
13. K. A. S. Quinn. Some Constructions for Key Distribution Patterns, *Designs, Codes and Cryptography* **4**, pages 177-191, 1994.
14. C. A. Shaffer. *A Practical Introduction to Data Structures and Algorithm Analysis*. Prentice Hall Press, 1997.
15. D. R. Stinson. On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption. *Designs, Codes and Cryptography* **12**, pages 215-243, 1997.
16. D. M. Wallner, E. J. Harder and R. C. Agee. Key Management for Multicast: Issues and Architectures. *Internet Draft (draft-wallner-key-arch-01.txt)*, <ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt>.
17. C. K. Wong, M. Gouda and S. S. Lam. Secure Group Communication Using Key Graphs. *Proceedings of SIGCOMM '98*, pages 68-79, 1998.

PKI Seeks a Trusting Relationship

Audun Jøsang¹, Ingar Glenn Pedersen², and Dean Povey¹

¹ Distributed Systems Technology Centre

Brisbane, Qld 4001, Australia

² The Norwegian University of Science and Technology

N-7491 Trondheim, Norway

Abstract. All human interaction is based on trust, meaning that we choose interaction partners and make commitment decisions based on how much we trust the other party. Digital certificates and public-key infrastructures represent an attempt to mimic real-world human assessment of identity and trustworthiness in an automated and mechanical fashion, but present implementations are based on a very limited trust model making them inadequate as a general tool for trust assessment and decision making. This paper describes public-key infrastructures in general and discusses issues related to trust management of public-key infrastructures.

1 Introduction

Public-key cryptography solves security problems in open networks but creates key management complexity. Digital messages can for example be signed by a private key allowing anyone with access to the corresponding public key to verify that the message is authentic, but this principle depends on the authenticity of public keys and the problem boils down to finding a method for secure distribution of public keys.

Public-key infrastructures (PKI) simplify key management and distribution but creates trust management problems. A PKI refers to an infrastructure for distributing public keys where the authenticity of public keys is certified by Certification Authorities (CA). A certificate basically consists of the CA's digital signature on the public key together with the owner identity, thereby linking the two together in an unambiguous way. In order to verify a certificate the CA's public key is needed, thereby creating an identical authentication problem. The CA's public key can be certified by another CA etc., but in the end you need to receive the public key of some CA out-of-band in a secure way, and various solutions can be imagined for that purpose.

However, there is a problem in this design. What happens if a CA issues a certificate but does not properly check the identity of the owner, or worse, what happens if a CA deliberately issues a certificate to someone with a false owner identity? Furthermore, what happens if a private key with a corresponding public-key certificate is leaked to the public domain by accident, or worse, by intent? Such events could lead to systems and users making totally wrong assumptions about identities in computer networks. Clearly CAs must be trusted

to be honest and to do their job properly and users must be trusted to protect their private keys. Trust management includes methods for assessing policies regarding issuance and handling of public-key certificates and for determining whether these policies are adhered to by CAs and users, with the purpose of making decisions related to on-line activities.

2 Public Key Infrastructures

Internet users can not base their judgements about identity of remote parties on faces or familiar voices, meaning that electronically received information a priori can not be trusted. Instead public-key certificates combined with public-key cryptography can be used to authenticate identity and message origin.

2.1 Certification Chains

The first certification system in any standard was the X.509v1 Authentication Framework[1], designed by CCITT/ITU for the purpose of securing the X.500 Directory[2] and intended to work with a hierarchy of certification authorities.

X.509v1 has been further developed into X.509v2 and X.509v3 in order to overcome weaknesses in the earlier versions, and X.509v3 is the basis of the IETF PKIX working group which is aiming at developing a general purpose public key certification infrastructure for the Internet. The format of X.509 certificates is illustrates in Fig.1.

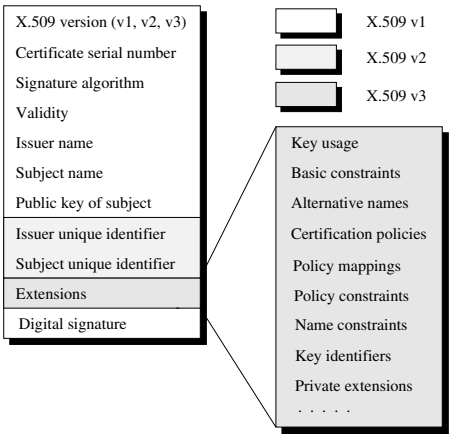


Fig. 1. X.509 certificate with extension part

The X.500 Directory was never adopted by the Internet community because it and the corresponding Directory Access Protocol (DAP) [3] were seen as too

complex for simple Internet clients to use. Instead LDAP (Lightweight Directory Access Protocol)[4] which is a relatively simple protocol for updating and searching directories running over TCP/IP is used.

The X.509 standard is often vague and open-ended, which means that an additional X.509 certificate profile is required in order to implement certificate handling on a real system. Since the original X.509 certificate format was specified to fit with the original X.500 directory which is no longer used, the format contains a number of more or less redundant fields. In the main body only “Validity”, “Public key” and “Digital signature” are really needed. “Issuer name” and “Subject name” are still used, but they are interpreted as general names rather than entries in a directory. Apart from that everything else is found in the extension part.

In order for two users to verify the authenticity of each others public keys it is sufficient that there exists a certification path between them. A certification path is an ordered sequence of certificates which together with the public key of the initial certificate in the path can be processed to obtain the public key contained in the final certificate in the path. The rules for certification path validation are quite complex depending on what extensions are in the certificate. See e.g. [5] for examples.

2.2 Certification Hierarchies

Certification between nodes is directed from the certifier to the owner of the certified key. Certification is unidirectional when an agent *X* certifies the public key of another agent *Y*, and bidirectional when the agents *X* and *Y* certify each others public keys. In the PKI jargon “*certifying a user*” means “*certifying the user’s public key*”. Chained certification can form different topologies.

Strict Hierarchy. Most commercial PKIs are strict hierarchies, as illustrated in Fig. 2, and most only consist of one or two levels. The certification paths go strictly from the top root CA, eventually via intermediate CAs, and down to users, where the users are assumed to be certified by the leaf nodes.

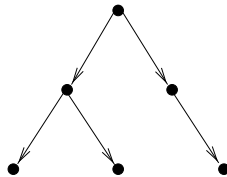


Fig. 2. A strict certification hierarchy

In a strict hierarchy all users can be easily identified and found because of the hierarchic structure. A user must know the public key of the top root in order to resolve certificate chains and establish a certification chain to any other user in the hierarchy.

General Hierarchy. A general hierarchy includes two-way certification between CAs, as illustrated in Fig. 3.

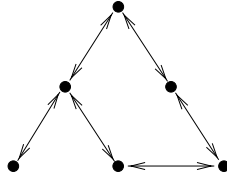


Fig. 3. A general certification hierarchy

When certification takes place in both upwards and downwards direction, each user will only need to obtain an authentic copy of the nearest CA's public key, while still being able to establish a certification path to every other user in the network. The X.509 standard [1] suggests a general hierarchy of this type, but no commercial PKI uses this topology.

Anarchic PKI. The opposite to a hierarchic structure is an anarchic structure where each CA (and user) is free to choose which other CAs (and users) it wants to certify, as illustrated in Fig. 4.

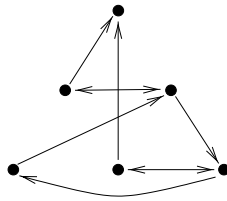


Fig. 4. Anarchic certification structure

The anarchic structure corresponds to the Web of trust on which PGP [6] is based. It consists of unidirectional and/or bidirectional certification between arbitrary agents. There is in principle no difference between users and CAs. The disadvantage of an anarchic certification network compared to a hierarchic structure is that there exists no simple algorithm for identifying certification paths between all users of an anarchic network, whereas such algorithms exist for hierarchic networks. A user must obtain as many public keys as possible in order to establish certification chains to other users.

Isolated Hierarchies. Many PKIs can exist in parallel without being linked to each other as illustrated in Fig.5.

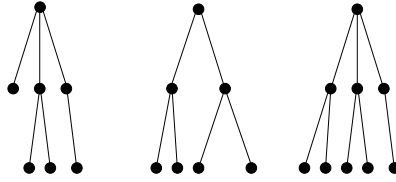


Fig. 5. Isolated certification hierarchies

A user must obtain the root public key of every PKI in order to verify certificates of users in all hierarchies. Users belonging to hierarchies with unknown root can not be identified.

PKIs used for the Internet Web consists of isolated strict hierarchies, and is in fact a topology of this kind. The root public keys are stored hard-coded in the most popular Web browsers. There are about 50 root keys delivered with Netscape Communicator release 4.6 or Microsoft Internet Explorer release 5.0.

Cross Certified Hierarchies. In order to avoid requiring that users acquire several public keys the hierarchies themselves can be cross certified as illustrated in Fig.6. In case the hierarchies are strict it is sufficient that a user obtains an authentic copy of the own root's public key while still being able to establish a certification chain to any user in any hierarchy.

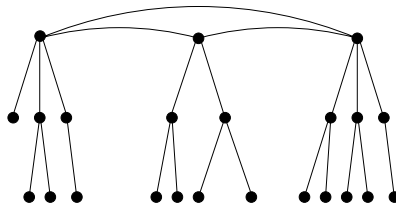


Fig. 6. Cross certified hierarchies

Cross certification between PKIs would simplify the distribution of root public keys, and would make PKIs truly open. The main problem opposing this development is that CAs in general have incompatible policies whereas cross certification requires some sort of policy alignment. A government can for example enforce a common policy and cross certification between all PKIs used by the public administration, but spontaneous cross certification between commercial PKIs has so far not been widespread.

3 PKI Management

The cryptographic aspects of PKIs are relatively well understood. The deployment of PKIs on the other hand requires management, and so far we have seen the emergence of two types of PKI management.

3.1 Web PKI

For Web PKIs the root public keys are hard-coded in Web browsers as *self signed* X.509v3 certificates, i.e. the public key has been certified by the corresponding private key. The only purpose of self certification is to simplify the certificate handling; the browser only needs to deal with certificates. Self certification provides no additional trust in the public key, and as such the term “self certification” can be misleading.

Since root certificates are hard-coded in the browsers they can not easily be upgraded. Root key management must in fact follow the pace of browser releases and distribution. Not only must changes be implemented in the next release of the most popular browsers, the users also have to upgrade the browser on their computers to the newest release. If for example public key revocation shall be useful it must be possible to enforce it relatively rapidly. Because this is not possible for root certificates it is in practice not possible to revoke them.

The most widely used application is presently to establish encrypted connections using the SSL protocol[7]. Another popular application is email encryption based on the S/MIME [8,9] standard which consists of digitally encrypting the body (and not the head) of email messages. A third application is for digitally signing SW components. The security problem users are facing regarding active components such as Java applets and Microsoft’s ActiveX components is whether such imported programs can safely be executed. One way this can be solved in Web browsers is to have the components digitally signed by the manufacturer’s public key which previously has been certified by a CA. This only indicates the SW manufacturer’s identity and does not say whether it is safe to let the SW component be executed.

3.2 Managed PKI

In contrast to Web PKIs, a managed PKI does not distribute root public keys piggy-backed with Web browsers, but is based on separate out-of-band procedures managed by the organisation that operates the PKI. This organisation usually operates CA servers from which user certificates can be down loaded. Managed PKIs are operated by an organisation to meet specific needs within the organisation or as a business activity. The organisation will have full control over the trust structure in the PKI hierarchy, but without being Web-born managed PKIs do not easily get global coverage. Managed PKIs can provide high trust and thus be suitable for high value transactions.

Organisations operating managed PKIs can decide to, or be enforced by law in a particular country, to establish cross certification to other managed PKIs and in that way create a PKI consisting of several interlinked certification hierarchies.

Secure distribution of the root public key is essential for managed PKIs, and a typical solution is to equip each user with a smart card containing the users private key in addition to the root public key.

4 Trust Management

It is assumed that trust is a belief based on knowledge, experience and perception. In the physical world trust in things and in other people is based on our experience with them, information we have received about them and how they appear to us. All this makes trust a very subjective phenomenon, meaning that I don't necessarily trust the same things or the same people as you and vice versa. The number of people we can potentially relate to within a physical world is also limited by distance and physical constraints. In the cyberworld on the other hand the number of people we can potentially relate to is only limited by the number of people that are on-line.

Cryptography can be interpreted as a mechanism for transferring trust from where it exists to where it is needed. For example if you initially trust the authenticity of a public key and you verify a message signed by the corresponding private key, then you will also trust the authenticity of the message. As such certificates and PKIs do not create trust, they merely propagate it, and users must initially trust something. Initial trust is traditionally established off-line, i.e. in the physical world, but it is perfectly possible to get experience and gain trust purely through on-line activities.

The challenge is to find a good method for making trust assessments about potential remote transaction partners in computer networks. A transaction partner can be someone you already know but it can also be someone who is totally unknown and with whom you have never interacted before and with whom you might never interact again after the transaction.

Trust assessment must be based on some initial trust combined with trust propagating mechanisms, and should provide a basis for decision making. Trust management is about all that, and can be defined as activity of making trust assessments by collecting, analysing and codifying relevant evidence with the purpose of making trust based decisions. The number of potential transaction partners on the Internet is presently around 100 million and the goal must be to make trust management schemes scalable to that size.

4.1 Certification Policies and Certification Practice Statements

The degree to which users can trust the binding between the public key and the owner identity stored in a certificate depends on several factors, including the practices followed by the CA in verifying the identity of the owner, the CA's operating policy, procedures and security controls, the owners obligations e.g. regarding secure storage of the private key, and legal obligations of the CA such as e.g. warranties and obligation limitations.

According to X.509 a certification policy is “*a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements*” [1]. To the degree that a certification policy exists or is applicable to a particular application it provides users with evidence for assessing the trustworthiness of certificate issued by CAs that adhere to the policy.

A more detailed description of the practices followed by a CA can be found in its Certification Practice Statement (CPS). According to the American Bar Association Digital Signature Guidelines “*a CPS is a statement of the practices which a certification authority employs in issuing certificates*” [10]. The CPS defines under what conditions certificates are issued and which liabilities the CA takes on itself and which are put on the user. Usually a CA offers different certification classes with varying degree of confidence in the key-to-owner binding with the purpose of being suitable for different applications. The general principle is: The higher the confidence, the more thorough the identity verification before issuing a certificate. For commercial CAs it can be added that the price you pay for a certificate increases with the confidence level.

The concepts of certificate policy and CPS were developed by different bodies for different reasons. A CPS is very specific and usually applies to a single CA whereas a certificate policy is more general and is intended to be applicable to larger domains.

Computers do not understand policies and CPSs although some attempts are being made for that purpose by including a certification policy extension in the X.509 certificate together with policy constraints and policy mappings. The certification policies extension in its minimal form provides a means of identifying the policy a certificate was issued under.

The idea behind the policy constraints and policy mappings is that the application can automatically check and enforce them at run time, for example by checking that the certificate policy corresponds to the application policy or that the certification policies of certificates in a chain are compatible. This of course requires standardised policies and constraints, and although seemingly difficult to achieve it would create several new possibilities.

4.2 Trust Management for the Web PKI

Unfortunately the notion of certificate policy does not exist for the Web. Primarily it would be rather difficult to define a policy that satisfies all user communities as well as all present and future applications, and secondly if such a

policy could be defined it would probably be too general to be useful to anyone. Instead each CA that operates on the Web has its own CPS.

If it shall be meaningful for a user to trust a PKI based on the trust in the root CA, then the CPS strength of intermediate CAs must be equal or increasing downwards in the hierarchy. In the same way, if it shall be meaningful to trust other cross certified PKIs based on the own root CA, then the CPS strength in other PKIs must be equal or greater. However, CPSs are usually difficult to compare unless they are accredited against an industry standard policy and can not be translated into a one-dimensional measure. It is therefore difficult to establish cross certification between PKIs, and as a result the Web PKI consists of isolated hierarchies, but because the public keys of all root CAs are hard-coded and distributed with the Web browsers cross certification is not really needed.

When accepting a certificate the user should really check the corresponding CPS, and if applicable the certification policy, to see what the certificate is worth, but because this would require the user to read a document of at least 10 pages each time a secure Web site is visited it is hardly ever done. The trust model of the Web PKI is more the result of the need for a business model than the need for good trust management.

The advantage for the major browser manufacturers is that they can provide CAs with global coverage by distributing the CA root public keys hard-coded in the browsers, the advantage for the CAs is that they can provide Web server operators with global acceptance of their server certificates, and the apparent advantage for the users is that they get a Web interface that anyone is able to use and that hides the complexities of the underlying cryptographic mechanisms.

A problem with this model is that distribution of root keys is linked to the distribution of particular Web browsers. As already mentioned the present implementation of PKIs in Web browsers does not allow certificate revocation and thereby represents a time bomb. If a private key is leaked to the public domain the corresponding certificate can be misused by anyone in order to set up false secure Web sites, send false email messages or distribute malicious SW under false name, and tragically there exists no simple way of stopping such attacks even after they have been reported. Another problem is that the Web interface provides very poor trust management to the user. In order to trust a secure Web site, an encrypted message or a digitally signed SW component only the identity of the certificate owner identity and the CA identity are available to the user. The user can only make an informed decision if both are known, but is left totally in the dark in case they are not. The browser interface usually presents dialog boxes but experience shows that users find dialog boxes annoying and tend to blindly click "yes" or simply turn off the security settings in order not to be interrupted by dialog boxes anymore. The following section describes potential consequences of these problems.

4.3 The SSL Attack

In order for a user to trust that something is authentic he or she must be presented with some evidence that can be correctly interpreted. For the purpose of

verifying the authenticity of Web servers SSL represents a strong authentication mechanism, but unless the evidence of this verification is intelligibly presented to the user the mechanism can be quite useless.

Assume a user A who wants to access Web services from secure server B which for example can be a bank providing financial transaction services for its clients. In the normal scenario, client A points his Web browser to bank B 's Web site. The Web server returns B 's certificate Cert_B to A 's browser which verifies the certificate using the pre-stored public key of the root CA that generated Cert_B . After successful certificate validation A 's browser continues the communication with B in secure SSL mode.

Fig.7 below shows user A 's client machine on the left and bank B 's server on the right side. We will show that the intruder T in the middle is able to make both A and B think they are communicating with each other although they in fact communicate with T .

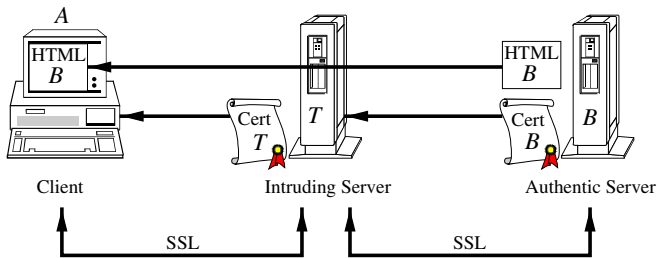


Fig. 7. Attack on SSL authentication

In the attack, the intruder Web server T acts as a relay between A and B passing the HTML pages from B to A and the requests from A to B . For the attack to work user A must be fooled into pointing her browser to T instead of to B . This can for example be done by placing a false URL on a portal until somebody accesses T from it in the belief that he or she accesses B . After a successful attack, the false URL can be removed in order not to leave any evidence of where the attack came from.

It is assumed that the intruder T has obtained a valid certificate Cert_T , either by buying it or because a private key with a corresponding certificate has been leaked to the public domain. When the client A has established a SSL connection to the intruding server T using T 's certificate, the intruding server establishes a SSL connection to the bank B using the bank's certificate Cert_B and simply relay the data sent by A and B to the opposite sides via two different SSL connections, including possible user passwords, so that A and B think they communicate with each other. When A sends a request to transfer money from her own account for paying a bill, T is able to modify the destination account number and the amount.

When a secure SSL connection is established the server is supposed to be authenticated by the client, as indicated by the key or the padlock icon on the browser window. However, this only indicates that something is authenticated and not what in particular, which for all practical purposes means that nothing at all has been authenticated. The blame for this vulnerability can of course be put on poor interface design, but the problem is also related to user awareness, and technology can only help making awareness easier to practice.

The browser does allow viewing a certificate by clicking on the padlock icon, but users hardly ever do this, and even security aware users who view the certificate when accessing a secure Web site can have difficulty in judging whether the information on the HTML page has been sent by the owner of the certificate. The system interface should make it easy to view, understand and believe in certificates. Requiring the user to explicitly make several extra mouse clicks in order to view a certificate that is hardly intelligible is simply not good enough.

A better user interface can be based on including the server company logo in the certificate and let the interface always display it when a secure connection is established. This will allow a quick visual check anytime without extra mouse clicks in order to verify the server identity. Alternatively an audible message can be included in the certificate and played to the user. However, users might not pay any attention to these features after a while, so that the problem of user awareness will not go away. All we can do is to create a system interface that helps users to be more aware.

5 A Better Trust Management

Some of the problems persisting in present PKI implementations include:

- **Computers don't understand the semantics of a policy.**

A certificate policy or a CPS is a piece of rather lengthy and complex prose text that can only be read by humans. It would be desirable to be able to specify certificate policies in such a form that they can be interpreted by computers in an automatic and mechanical fashion.

- **Cross Certification Requires Equal Policies.**

If it shall be meaningful to trust the root CA in a PKI as a representative of cross certified PKIs then the policy strength must equal in both PKIs. However, "policy strength" is a multi-dimensional measure that is difficult to match. Cross certification is therefore difficult to achieve unless the policies themselves are equal.

- **PKIs do not handle trust dilution.**

Real-world trust is intuitively diluted in a chain of recommendations, but present PKI implementations only provides a binary trust model.

- **PKIs do not take into account parallel certification paths**

Real world trust is often based on multiple recommendations. It would be useful to have something similar in PKIs but present implementations only handle single certification chains.

– **PKIs give little support for decision making.**

PKIs give little support for answering questions such as e.g.: “*For a given certification path and a given user what is the upper transaction value you are willing to risk?*” and “*For a given transaction how can you select the transaction partner that will minimise your risk?*”.

5.1 Including Subjective Trust Measures in Certificates

One possible solution is to include subjective measures of trust within a certificate and combine these parameters in order to deduce trust in remote users and systems. Such a trust measure can represent the policy within the certificate in addition to trust in the reliability of CAs and users themselves, can be understood by humans and can be automatically handled by computers. In addition, trust dilution can be handled by combining trust measures in series and trust from multiple certification paths can be combined in parallel. It would allow cross certification can be established between arbitrary pairs of CAs by simply specifying within the certificate how much one CA trusts the other. Finally trust measures can be combined with transaction utility functions in order to support decision making.

This type of trust model is for example described in [11]. Unresolved problems related to this model is how users can consistently determine subjective trust measures. Another problem is that the trust measures are communicated to other users and thereby can not be kept confidential whereas a CA might not want to disclose that it distrusts users or other CAs. Finally CAs might not want to make any explicit statement about trust in other CAs as it might imply liabilities.

5.2 Policy Alignment

A common misnomer with X.509 based PKIs is that CAs are making statements about trust when issuing a certificate. In most cases, a CA is merely following a defined procedure and evaluating objective evidence to determine a binding between a public key pair and the issue of whether to trust this user is one that the relying party must resolve, based on their subjective judgements about the statements and policy made by the CA. This is particularly important in the case of certification chains, which are usually not “chains of trust” as some PKI literature suggests, but are in fact simply a series of signed statements by CAs that attest to a binding between keys and some objective evidence as determined by following the process outlined in a particular policy and CPS. It is believed that the commercial nature of most CAs precludes them from making subjective statements, as this exposes them to too much risk. Contrast with the PGP system which is based on meeting the needs of small communities of users, and where users are much more likely to make statements about trust.

One of the biggest problems for users when evaluating policy chains is the problem of ensuring equivalent policies across all links of the chain. The approach taken by X.509 is to specify a path validation algorithm that requires that all

certificates in a chain contain the same policy identifier, or a policy that has been “mapped” to an equivalent policy using the `policyMappings` extension. This approach reflects the general philosophy of X.509, which tends towards a centrally managed design. This is also consistent with related work such as the NIST PKI architecture[12] and work being done in the Australian Standards working group IT/12/4/1 to develop an architecture for PKI systems [13]. Both these approaches have an architectural model which consists of a hierarchy of components consisting of the following:

- **A Policy Approval Authority (PAA).** This is a root authority whose job is to approve the policies of other entities in the hierarchy. Note that the PAA itself does not assert any policy, it simply issues a self-signed certificate that must be trusted by some out of band means. The PAA will most likely be a statutory body created by a government and will operate according to a particular set of rules.
- **A Policy Creation Authority (PCA).** The role of the PCA is to act as a policy creation body of a community of users. An example would be a group of banks who get together to decide on a common policy for retail/wholesale banking. These banks would form a single entity – a banking PCA, that was responsible for setting policy in the banking sector. The PAA would then approve the policy(s) set by this PCA by issuing it with one or more certificates asserting the policy identifier.
- **Certification Authorities.** The CAs will issue certificates under one or more policies. These policies may have been set by a PCA, or they may be specific to the CA (either with an approved policy – i.e. the CA is acting as a PCA, or an unapproved policy – i.e. the CA is operating outside the architectural model).

This paradigm encourages users to trust the PCA to set an appropriate policy for the domain of application. The user simply selects the policy identifier defined by the PCA for the given application and trusts that the infrastructure and requirements set up by the PCA and PAA are such that the policy will be appropriate, and that CAs operating under that policy will behave appropriately. In a sense the user is deferring some of their trust to the infrastructure. This is definitely appropriate in many cases, we tend to rely on our legal, and socio-political systems to a large extent when conducting commerce in the real world, so it only seems natural to translate this into the electronic world.

However, there are also a number of problems with this model. Many governments (including Australia’s) have punted on the idea of setting up a PAA. This seems largely due to the perception of this body as being a huge liability risk due to the large amount of damage possible from compromising the PAA’s private keys. The absence of a PAA favours a disjoint model, with a number of PCAs acting as root authorities for their community of interest, and requiring the user to obtain the public keys/self-signed certificates of each of these CAs by out-of-band means.

In addition, it makes the issue of cross-certification quite difficult. In order to cross-certify, two CAs must agree to mark each other’s policies as equivalent.

However, it is more than likely that this will not be the case. X.509 contains a number of mechanisms for parent CAs to prevent sub-ordinate CAs from doing this. They can inhibit the ability for child CAs to do policy mapping, and they can restrict the certification path such that a subordinate CA can only issue certificates to end-entities and not to other CAs. It is more than likely that cross-certification would have to occur at the level of PCAs to enforce the policy. This can also be used to enforce commercial limitations on CAs. For example, a commercial CA with its root keys embedded in the browser sells a cross-certification service so that another CA can issue certificates that do not require browser bootstrapping. However, this CA is unlikely to want to allow the subordinate CA to then resell this service to other CAs and so will seek to limit the subordinate CA to only issuing end-entity certificates.

Another criticism is that while a single policy identifier is enforced across multiple CAs, there are clearly different roles being played by each certificate in the chain. For example:

- The PAA will issue a self-signed certificate containing its public key. As noted this is simply a convenient mechanism for “bootstrapping” the certification path, and therefore asserting a policy in this certificate is basically meaningless.
- The PAA will then issue a certificate to the PCA asserting the policy identifier which the PCA has created. This certificate is a statement in effect that the PAA “approves” this policy for use within its framework.
- The PCA will then issue a certificate to one or more CAs asserting the policy identifier it has created. This certificate is a statement that the CA identified is authorised to operate within the community of interest that the policy defines.
- The CA issues a certificate to an end-entity. This certificate is a statement that the CA has followed the rules of the policy as they relate to registration of users, and has verified the binding between the user and their public key.

While these roles are very different, they are essentially implicit semantics, and the only thing that the user knows is that the certificates have all been issued under the same/equivalent policy. The user must simply trust that the framework that exists has ensured that the ordering of CAs in the chain is appropriate to their function.

While this may be appropriate in some instances, it makes it difficult to build ad hoc relationships, and it means that information is lost in the process. This is particularly problematic in the case of cross-certificates.

What is needed is a way for the information about policies to be expressed in a standard interpretable way such that a relying party can make all the decisions about whether policies are equivalent, and whether they are appropriate for a given application. It is after all, the relying party who has to bear any risk about who/what they trust.

6 Conclusion

A public-key infrastructure is technically seen a structure of cryptographically linked data objects that can be transmitted through a data network and from which public keys can be extracted. However, extracting public keys and using them in security applications is only meaningful if the public-key infrastructure is linked to a corresponding trust structure, and for this purpose trust management is needed. Trust management can be defined as the activity of making trust assessments by collecting, analysing and codifying relevant evidence, with the purpose of making trust based decisions. Trust models in present PKI implementations are too limited for providing users with trust management facilities, and possible solutions include more rich trust models and standardised certificate policies.

References

1. ITU. *Recommendation X.509, The Directory: Authentication Framework* (also ISO/IEC 9594-8, 1995). International Telecommunications Union, Telecommunication Standardization Sector (ITU-T), 1993.
2. ITU. *Recommendation X.500, Data Communication Network Directory*. International Telecommunications Union, Telecommunication Standardization Sector (ITU-T), 1993.
3. ITU. *Recommendation X.509, The Directory: Abstract Service Definition* (also ISO/IEC 9594-3, 1995). International Telecommunications Union, Telecommunication Standardization Sector (ITU-T), 1993.
4. M. Wahl, T. Howes, and S. Kille. *RFC 2251 - Lightweight Directory Access Protocol Version 3 (LDAPv3)*. IETF, December 1997.
5. R. Housley, W. Ford, W. Polk, and D. Solo. *RFC 2459 - Internet X.509 Public Key Infrastructure, Certificate and CRL Profile*. IETF, January 1999.
6. P.R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.
7. Netscape. *The SSL 3.0 Protocol*. Netscape Communications Corp, 1995.
8. S. Dusse. *RFC 2311 - S/MIME Version 2 Message Specification*. IETF, March 1998.
9. S. Dusse. *RFC 2312 - S/MIME Version 2 Certificate Handling*. IETF, March 1998.
10. ABA. *Digital Signature Guidelines: Legal Infrastructure for Certification Authorities*. American Bar Association, 1995.
11. A. Jøsang. Trust-based decision making for electronic transactions. In L. Yngström and T. Svensson, editors, *Proceedings of the Fourth Nordic Workshop on Secure Computer Systems (NORDSEC'99)*. Stockholm University, Sweden, 1999.
12. W. Burr, D. Dodson, N. Nazario, and W.T. Polk. Minimum Interoperability Specification for PKI Components, Version 1. Technical report, NIST, September 1997.
13. Standards Australia. *PKAF Report: Strategies for the implementation of a public key authentication framework (PKAF)*. WG IT/12/4/1, 1996.

The PKI Specification Dilemma: A Formal Solution

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

Information Technology Division
Defence Science and Technology Organisation
PO Box 1500, Salisbury
SA 5108, Australia
{Maris.Ozols, Marie.Henderson, Chuchang.Liu,
Tony.Cant}@dsto.defence.gov.au

Abstract. In this article we look at some of the problems associated with trying to provide a specification for a Public Key Infrastructure (PKI). We focus on the PKI profile that has been proposed for use with the Internet, known as PKIX (PKI using X.509 certificates), to provide a specific example. Our intention is to add value to this and other specifications by providing a more formal description and a framework in which to develop path validation procedures. We take the path validation algorithm in PKIX and give a formal description of the actions and properties it defines. In this way, the essential aspects can be captured and clearly formulated, which would facilitate the testing of implementations in a more rigorous and well defined way.

1 Introduction

There is a global movement towards using public key technology within a Public Key Infrastructure (PKI) to support secure digital communications (for example: electronic commerce and secure messaging). Details on public key cryptosystems can be found in [19] or [4]. In order to use public key cryptography, it is essential to make the public key available to those one wishes to communicate with in a trusted way. Certificates are created by Certificate Authorities (CAs) and are used as the vehicle to transport public key details. The CA attests to the correctness of the information in the certificate by signing the certificate (this also protects the information from being altered thereafter). The certificate is verified by checking the signature, in which case a valid copy of the public key of the CA is required. In a large PKI, multiple CAs may be required with chains of certificates and even possibly multiple trust points, i.e., multiple trusted public keys.

The Internet Engineering Task Force (IETF) is involved in guiding the evolution of the Internet and is the main organisation involved in developing Internet Standard specifications. Agreement on content and meaning is arrived

* Current address: Information Security Research Centre, Queensland University of Technology, Brisbane QLD 4001, Australia. E-mail: marie@isrc.qut.edu.au.

at through a process involving both informal and formal discussions along with Working Groups (WGs) and a series of draft publications from which the final Request For Comment (RFC) is distilled. The process is sometimes described as forming a “rough consensus” on the appropriate content and meaning. The RFCs that form part of an effort towards standardisation need to be clear as well as being inclusive and accessible requirements that are often at odds. This type of problem also arises with most other standardisation efforts.

In this article we are interested in the PKI profile outlined in RFC 2459, which is concerned with standardising a Public Key Infrastructure (PKI) for the Internet, see [6]. The PKI described in that document is known as PKIX as it uses the ITU-T X.509 version 3 (X.509) certificate format [20]. Our main focus is the path validation protocol from Section 6 of [6], which consists of an algorithm used as a specification in the following way: “implementations are required to derive the same results but are not required to use the specified procedures”. Such a specification could lead to varying definitions or interpretations of conformance. In fact, there are already a number of problems, such as those we mention below, arising from this approach.

The English language is wonderfully expressive: a single word or phrase is able to convey many different meanings. This is why it is unsuitable on its own for providing a precise specification. When only a natural language specification is given, it is probable that there will be different interpretations which all meet the specification, although they may logically be different. General examples of this type of problems are widespread. Such problems should be considered seriously for documents that are expected to become global standards involving the transfer of trust. In mid 1999 a defect report [3] was filed concerning the X.509 standard. A number of defects were identified, some of which were serious and could cause incorrect acceptance of certification paths. Due to the wide acceptance of X.509 this affects a number of other standardisation efforts, like PKIX. The PKIX WG is working on a new version of RFC 2459 (this draft is available from <http://www.imc.org/draft-ietf-pkix-new-part1>), hoping to anticipate the changes to X.509. The PKIX document is therefore important as it indicates the likely direction for all standards relying on X.509.

It has also been recognised that interoperability problems have occurred with S/MIME products, see [11]. It is argued that the looseness of the S/MIME specification has led to the development of a number of compliant but nonetheless non-interoperable products. It is likely that such problems will occur with PKIX as well, the extent of which will be realised as products being to appear. Although example conforming certificates are given with PKIX there are no example conforming certification paths. It could be argued that example certification paths would provide such narrow coverage of the possible cases so as not to warrant inclusion. Whether the reader agrees with this or not it is clear that the absence of examples makes a clear specification even more important. Most specifications of the IETF are supported by descriptions in natural language, as well as more formal descriptions and examples. This approach is supported by the IETF document guidelines. Requiring technical knowledge of the audience is largely

unavoidable in these areas, thus making the exclusion of technical descriptions less relevant.

Experience with implementing PKIX path processing, in particular with regards to the more exotic parts such as policy and constraints, is not yet well developed. Due to the slower than anticipated progress in PKI development, most implementations can, at present, ignore these more difficult areas and focus on basic path processing. However, in this case they do not necessarily meet PKIX compliance requirements. As an example, Cryptlib [2] does implement the policies and constraints extensions. The following comment is taken from the cryptolib code: “Policy constraints are the hardest of all because, with the complex mishmash of policies, policy constraints, qualifiers, and mappings it turns out that no-one actually knows how to apply them.” Perhaps the PKIX specification is silent in areas where it should not be.

As is well known, formal methods can provide strict proof technologies for verifying critical properties of a 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. In this article we take the PKIX path validation algorithm and give a formal description of some of the actions and properties it defines. In this way, the essential aspects of the certification path validation can be captured and clearly formulated, which would facilitate the testing of other implementations in a more rigorous and well defined way. A formal description of precisely what is required for conformance could also assist implementers during the design phase. We are not suggesting that our specification replace the one already given in RFC 2459 but that it may be used to improve clarity. As part of our examination, a framework for path validation is developed that would be applicable to PKIs in general.

The paper is organised as follows. In Section 2 we provide a brief background to the PKI elements described in RFC 2459 relevant to our discussion. In Section 3, we present a framework for path validation, in which three categories of path validation checks are identified. Section 4 discusses formalisation of the path validation algorithm in Isabelle/HOL, and presents the essential theories used for path validation. In particular we provide a framework function to implement those interactive calculations in a path validation procedure, and propose a formal technique applied for modelling PKIX path validation. Section 5 talks about the benefits of our approach. The last section concludes this paper with a brief discussion about possible future work.

2 Preliminaries

We do not give general descriptions of PKIs, certificates, Certificate Authorities (CAs) or the certification process. For these we refer the reader to Ford and Baum [4]. We also assume that the reader is familiar with the use of certificates to convey public keys, especially those associated with digital signature services. PKIX itself contains a description of its relationship to different PKIs such as Privacy Enhanced Mail (PEM) and use of earlier versions of X.509.

A detailed description of the certificate fields used by PKIX is given in [6] but we include a short description of the fields of interest for the convenience of the reader.

A X.509 certificate is a sequence containing the following three fields:

- **tbsCertificate**: a sequence of further fields over which the signature is calculated;
- **AlgorithmIdentifier**: identifies the public key algorithm which has been used to sign the **tbsCertificate**; and
- **SignatureValue**: the actual signature calculated over the **tbsCertificate**.

The **tbsCertificate** itself contains the following sub-fields of interest to our discussion:

- **Version**: the version of the X.509 certificate format being used – v3 with PKIX;
- **Signature**: same as the **AlgorithmIdentifier** above;
- **Issuer**: identifies the *issuer* of the certificate;
- **Validity**: gives the time period for which the certificate is valid;
- **Subject**: identifies the *subject* of the certificate;
- **SubjectPublicKeyInformation**: gives the subject's public key value and identifies the public key algorithm with which it is meant to be used; and
- **Extensions**: contains a sequence of extension fields.

Before making a further discussion about the field **Extensions**, we give a definition related to certification path validation as follows:

A certification path is a non-empty sequence of certificates $\langle C_1, \dots, C_n \rangle$, where C_1 is a trusted certificate, C_n is the target certificate, and for all i ($1 \leq i \leq n-1$) the subject of C_i is the issuer of C_{i+1} . The trusted certificate is viewed as a certificate that is trusted by the verifier and it should of course be a valid certificate, and the target certificate is the one that the verifier wants to verify. The certificate validation procedure takes a given certification path and determines whether the target certificate is valid or invalid.

The problem of how a certification path is obtained can be solved easily, see [8]; it will not be considered in this article. We will focus on the path validation algorithm.

Extensions are used to automate the path validation process. Each extension field can be marked as *critical* or *non-critical*. Any extension can be ignored, but in the case of critical extensions, the issuing CA will take no responsibility for use of the certificate. Note that an Object Identifier (OID) is a unique sequence of positive integers used to distinguish objects. Each extension is associated with an OID defined in X.509. The PKIX extensions that are important to our discussion are:

- **CertificatePolicies**: contains a sequence, possibly empty, of policy OIDs. The purpose, as stated in PKIX, is that in a certificate these policy information terms are used to indicate the policy under which the certificate was issued and for which the certificate may be used;

- **PolicyMapping**: is used in CA certificates to link separate certificate domains by providing a mapping of policies of one domain to policies of another domain.
- **PolicyConstraints**: consists of two further fields:
 - (1) **InhibitPolicyMapping**: a positive integer which indicates how many additional certificates may occur in a path before policy mappings are no longer allowed;
 - (2) **RequireExplicitPolicy**: a positive integer which indicates how many additional certificates may occur in a path before an acceptable policy OID will be required in each certificate.

An application is expected to have its own list of acceptable policy OIDs by which it can compare with the list of policy OIDs in a certificate. Some applications may not require any particular policy at all.

3 A Framework for Path Validation

The framework for certification path validation is based on the certificate format. As discussed in the previous section, a X.509 certificate consists of three fields and some fields further contain several sub-fields. Without loss of generality, we may simplify our discussion by assuming that the PKI certificates have a standard public-key certificate format proposed in [7], which contains the following basic information: the name of the certificate issuer, the start and expiry dates, the subject (i.e., the name of the holder of the private key for which the corresponding public key is being certified), the value of the public key, the extension field, and the signature of the issuer. Formally, a certificate has the following form:

$$\mathbf{Cert} \langle \mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E}, \mathbf{SIG} \langle \mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E} \rangle \rangle$$

or, simply, written as $\mathbf{Cert} \langle \mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E}, \mathbf{SIG} \rangle$, where $\mathbf{I}$ is the issuer, $\mathbf{D}_s$ and $\mathbf{D}_e$ are the start date and expiry date respectively, $\mathbf{S}$ is the subject of the certificate, $\mathbf{PK}$ is the value of the public key held by $\mathbf{S}$, $\mathbf{E}$ is the value of the extension field, and $\mathbf{SIG}$ is the signature of the issuer $\mathbf{I}$.

Given a certificate $\mathbf{C} = \mathbf{Cert} \langle \mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E}, \mathbf{SIG} \langle \mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E} \rangle \rangle$, the following projection functions can be used to obtain the value of each component contained in $\mathbf{C}$:

$$\begin{array}{lll} \overline{\mathbf{I}}(\mathbf{C}) = \mathbf{I} & \overline{\mathbf{D}}_s(\mathbf{C}) = \mathbf{D}_s & \overline{\mathbf{D}}_e(\mathbf{C}) = \mathbf{D}_e \\ \overline{\mathbf{S}}(\mathbf{C}) = \mathbf{S} & \overline{\mathbf{PK}}(\mathbf{C}) = \mathbf{PK} & \overline{\mathbf{E}}(\mathbf{C}) = \mathbf{E} \\ \overline{\mathbf{SIG}}(\mathbf{C}) = \mathbf{SIG} \langle \mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E} \rangle & \mathbf{tbs}(\mathbf{C}) = (\mathbf{I}, \mathbf{D}_s, \mathbf{D}_e, \mathbf{S}, \mathbf{PK}, \mathbf{E}). \end{array}$$

In the following, we introduce a path validation framework, that makes important and natural distinctions between the types of checking done for validating certification paths. Certificate verification is intended to prove that the certificate is valid and no-one has tampered with the contents of a certificate.

This includes verifying the identity of the issuer and owner, verifying the signature, and checking the associated revocation listing. To verify the signature, the validity of the issuer's certificate needs to be determined, and so on, until a trusted certificate is encountered. Therefore, certificate verification involves iteratively verifying certificates in a certification path from a trusted certificate. The best method for obtaining the certification path is dependent on the structure of the PKI. The PKIX path validation algorithm takes the path as an input, so the exact means of certification path discovery is not important for this discussion.

In regards to path validation, all the checks of certificates can be separated into the three main categories listed below. In our framework, all of the certification path validation checks can be placed into one of the three categories.

Single checks: these are checks that are performed on and only involve a single certificate (as is the case with the **Validity** field, for example).

Pair checks: these checks require the comparison of two possibly different fields from two certificates. For example, the **Issuer** field of the certificate C_{i+1} and the **Subject** field of the certificate C_i (where $1 \leq i \leq n-1$) need to be the same.

Path checks: for these checks the entire certification path $\langle C_1, \dots, C_n \rangle$ may be required. If the certificate C_i holds the value k in the **requireExplicitPolicy** field of the **PolicyConstraints** extension then it must be checked that the certificates C_{i+k} onwards have an acceptable policy in their **Policy** extension fields and concurrently checked whether the value k is decreased by any certificates further along the path.

This framework is the basis for a natural separation of the entire certification path validation problem into distinct types based on the different checking requirements. Actually, there are several implementations that follow this, although usually not emphasised in the documentation.

4 Formalisation in Isabelle

Isabelle [13,15] is a generic theorem prover that can be used for implementing a range of logical formalisms. The structure and behavior of a certificate verification system can be formalized in an appropriate logical formalism [7], so it becomes possible to mechanize a PKI algorithm in Isabelle. We use Isabelle/HOL, an instantiation of Isabelle relying on higher-order-logic (Isabelle can be instantiated to various base logics). The essence of higher-order logic is that both functions and predicates (of the appropriate type) can take functions or predicates as arguments, and return them as results [5].

In the following, we present the essential theories that are used for path validation.

4.1 Certificate Theory

In Isabelle, a theory consists of the definition of types, functions including the fact that functions can be constants, and rules/axioms. Figure 1 gives an Isabelle theory, named `Certif.thy` or, simply, `Certif`. This theory defines the data type of certificates, and includes several rules which express the projection functions in the Isabelle theory.

```

Certif = Main +
datatype
  ('n, 'd, 'k, 'e, 's) cert = "Cert" 'n 'd 'd 'n 'k 'e 's
consts
  issuer      :: "('n, 'd, 'k, 'e, 's) cert => 'n"
  subject     :: "('n, 'd, 'k, 'e, 's) cert => 'n"
  start       :: "('n, 'd, 'k, 'e, 's) cert => 'd"
  expire      :: "('n, 'd, 'k, 'e, 's) cert => 'd"
  public      :: "('n, 'd, 'k, 'e, 's) cert => 'k"
  extensions  :: "('n, 'd, 'k, 'e, 's) cert => 'e"
  sig         :: "('n, 'd, 'k, 'e, 's) cert => 's"
  tbs        :: "('n, 'd, 'k, 'e, 's) cert => ('n * 'd * 'd * 'n * 'k * 'e)"
rules
  issuer_def   "issuer      (Cert I ds de S PuK exts SIG) = I"
  subject_def  "subject     (Cert I ds de S PuK exts SIG) = S"
  start_def    "start       (Cert I ds de S PuK exts SIG) = ds"
  expire_def   "expire      (Cert I ds de S PuK exts SIG) = de"
  public_def   "public      (Cert I ds de S PuK exts SIG) = PuK"
  extensions_def "extensions (Cert I ds de S PuK exts SIG) = exts"
  sig_def      "sig         (Cert I ds de S PuK exts SIG) = SIG"
  tbs_def      "tbs c = (issuer c, start c, expire c, subject c,
                        public c, extensions c)"
end

```

Fig. 1. The Certificate Theory

In this theory, “`Certif = Main +`” declares that `Main` is the parent theory of the theory `Certif`. `Main` is a basic theory in Isabelle/HOL, which collects all the basic predefined theories of arithmetic, lists, sets etc. Hence `Certif` is built upon the basis theory by defining a new data type (`Cert`), new syntax for functional constants (e.g. `issuer`), and rules that give the definitions and properties about these functions (e.g. `issuer_def`).

With the theory `Certif` defined, we can prove some goals. For example, the following goal can trivially be proved by rewriting with the new rules above:

```
Goal "tbs (Cert I ds de S PuK exts SIG) = (I,ds,de,S,PuK,exts)";
```

Once a goal is proved, it can become a theorem of the `Certif` theory, and may be used in later proofs.

4.2 Modelling Cryptographic Functions

In order to model the cryptographic functions that are involved in the public key certificates, we adopt the following principles:

- *Higher-order, property-based approach.* That is, rather than define a primitive function for signing, and another for checking signatures, we define the types and relationships that such a pair of functions must satisfy. This has the advantage of making it easy to extend the theory (and results) for multiple algorithms. It is also logically more secure, since if there are axiomatically defined functions which are inconsistent, then the theory will be unsound, while if we make the corresponding error in the property relating signing and checking, then the logic remains sound, although the property becomes uninteresting as no functions can satisfy it.
- *Purely looking at the essential properties.* In the case of our cryptographic functions, we focus on the essential connection between signing and checking, and their relationship to the public and private keys. We do not model the signing mechanism itself, so our theory does not include any discussion of encryption, and hence does not reflect the idea of signature as encryption with a private key.

Based on the above principles, we construct the signature theory named as `Sign.thy` shown in Figure 2.

```

Sign = Main +
consts
  is_sign_pair :: "('m => 's) * ('m => 's => bool)) => bool"
  is_PK_sign_system :: " (('x => 'm => 's) * ('y => 'm => 's => bool) *
                        (('x*'y) => bool)) => bool"
rules
  sign_pair_def  "is_sign_pair (sign, check) =
                  (!M S. check M S = (S = sign M))"
  PK_sign_system_def "is_PK_sign_system (sign, check, keyPair) =
                    (! X Y. keyPair (X, Y) --> is_sign_pair (sign X, check Y))"
end

```

Fig. 2. The Signature Theory

The signature theory includes two rules:

- The first rule (`sign_pair_def`) indicates that checking a message `M` against a signature `S` should succeed in precisely those occasions when `S` equals the result of signing `M`.
- The second rule (`PK_sign_pair_system_def`) means that the signing operation formed from private `X` can be checked by the checking operation formed by `Y`, whenever `X` and `Y` represent a valid key pair.

Our approach is quite general. However, in order to accommodate DSA [12], we would need to extend our model to allow non-deterministic parameters. Note also that we allow the types of the private and public keys to be different, in contrast to much work that has been done in the authentication protocol verification area, such as Paulson [14], Burrows *et. al.* [1], Roscoe [18] and Lowe [9] etc.

4.3 Framework Function

As already described, the various checks used for path verification can be grouped according to what data they are calculated on. We now need to describe in Isabelle a *framework function* that may take all these individual checks, and apply them to a certification path to be verified.

In the case of single and pair predicates, it is clear that these may be iteratively applied to the list of certificates. The results of these tests are joined by conjunction (since if any test fails the entire path must fail to validate). So the pattern of iteration can be described by:

$$B_{i+1} = B_i \cap (\text{pair}(C_i, C_{i+1})) \cap (\text{single } C_{i+1})$$

where B_i is the accumulating boolean result (true meaning “valid so far”), and C_{i+1} is the current certificate we are processing (the framework function is defined so that the previous certificate processed is made available to the function `pair`).

The remaining functions can depend on any aspects of the list, and hence are the most general, and potentially worst behaved (from an ease-of-modelling viewpoint). However, in all the cases we have seen in path validation algorithms, there is a fairly simple sequence of iterative structures G_i that can be constructed down the certification path that allows these predicates to be calculated. Examples will be briefly considered later, but at the moment we ignore these, and extend our accumulating boolean result to cover also these last predicates, represented by `path_pred` in the following:

$$\begin{aligned} B_{i+1} &= B_i \cap (\text{path\_pred}(G_i, C_{i+1})) \cap (\text{pair}(C_i, C_{i+1})) \\ &\quad \cap (\text{single } C_{i+1}) \\ G_{i+1} &= f(G_i, C_{i+1}) \end{aligned}$$

We introduce the framework function to implement these iterative calculations, using the various predicate arguments passed to it. It is called as follows:

```
framework path_state path_pred pair single state_init trusted_cert certs
```

where the meanings of arguments are as follows:

- `path_state` – the function that calculates the new state from the old state and the new cert (certificate). Needed for path properties that use the whole path.

- **path_pred** – The path property calculating whether the validation of the whole path (up to here) has failed yet.
- **pair** – pred for checking neighbours in the certificate chain.
- **single**: pred for checking single certs (e.g. for time).
- **state_init** – the initial values in the state used for whole path pred. Often either a “null” of some type, or derived from the initial trusted certificate.
- **trusted_cert** – the “top cert” that is trusted by the verifier.
- **certs** – an argument to the resulting validation function - the list of certs to be verified. Note that the head of **certs** is the highest in the tree.

4.4 Modelling PKIX Path Validation

PKIX path validation is quite complex, so we do not aim to yet model all aspects. Separate consideration of the processing regarding various extensions should be a valid technique. The reason is that most processing of extensions is independent of each other. It is, however, useful to look at interactions when they occur, and this would be a good area for further study.

The aspects we will be focusing on are the PKIX processing rules for dealing with “certificate policies”. These are defined as:

A named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with a common security requirement [6].

Informally we interpret the role of certificate policy processing in PKIX as determining for the users the set of policies under which a given certificate may be validly used. A given certificate may have been issued under policies P_1 and P_2 , but it may be that it is only valid for use under policy P_2 (perhaps because of restrictions applied by CAs higher in the path, or by the application carrying out the validation).

Closely related to certificate policies is the *policy mappings* extension, which is intended to associate related policies from different domains when cross-certifying between them. Although they play an important role in the PKIX algorithm, and have a major part in the discovered flaws, we do not yet include them in the modelling presented in this paper.

Consider now the predicates for checking certificate policies in PKIX. Our accumulating structures G_i need to track various information: the set of policies acceptable to the user (user policy set: **ups**), the set of policy constraints from CAs earlier in the certification path (authority policy set: **aps**), a counter for the number of certificate we are up to in processing (**cnt**), and the certificate number at which certificate policies must be present (policy *limit*: **lim**). These are grouped together in a tuple: (**ups**, **aps**, **lim**, **cnt**). In the following, we consider the path predicate for path predicate checking the policies of the next certificate (c) using the data of this tuple.

Given the current values for (**ups**, **aps**, **lim**, **cnt**), we use a particular path predicate named as **path_pred_cp** (cp for certificate policy) to process the next certificate, say **C**. The path predicate is defined in the following form:

`path_pred_cp_def` `"path_pred_cp((ups, aps, lim, cnt), C) = ..."`

where the detail of the right hand side of the equation is not given yet.

We now consider how the right hand side in the above definition is to be defined. From PKIX we have the following process in the path validation algorithm:

“(e) (1) if the certificate policies extension is marked critical, the intersection of the policies extension and the acceptable policy set shall be non-null.”

This can be expressed in the path predicate provided to the framework as follows:

$$(((\text{certPoliciesCritical} \wedge (\text{extensions } C)) \rightarrow (\text{aps} \cap (\text{certPoliciesExt}(\text{extensions } C)) \neq \emptyset)))$$

where $\emptyset$ is the empty set.

The following check is also done by the PKIX algorithm:

“(d) (1) if the explicit policy state variable is less than or equal to i , a policy identifier in the certificate shall be in the initial policy set.”

The algorithm has the following representation in Isabelle:

$$((\text{lim} \leq \text{cnt}) \rightarrow (\text{ups} \cap (\text{certPoliciesExt}(\text{extensions } C)) \neq \emptyset))$$

The two preceding predicates are conjoined together to complete the definition of `path_pred_cp` and are used in the framework. Also defined and used are functions that define how the various components `lim`, `cnt` and `aps` are updated, and these functions can again be directly linked to clauses in the PKIX definition.

Once a theory describing path validation is defined (in our case for a subset of PKIX path validation) it becomes possible to define paths of certificates, and to reason about certificate validity. In this way we can exhibit particular paths that illustrate unexpected behaviour of a given path validation algorithm. It will also be possible to prove that a particular algorithm may have certain desirable security properties, or satisfies a given security policy, though we have not yet carried out such proofs.

5 Benefits of Our Approach

Our approach can be used to detect or highlight inconsistencies and errors in standards and/or implementations. One of the ways that this is achieved is by the type-checking Isabelle carries out on the theory, which can reveal basic kinds of inconsistencies.

Formal specification may also be used to compare various alternatives to fixing a particular problem, and potentially to verify that some alternatives are

indeed free of error. For example, we are now looking at the suggested changes to PKIX, to see whether they are indeed free of some of the undesirable properties.

One of the major issues is that many standards, such as PKIX and X.509, are effectively specifying a *solution* (or a class of solutions). Although we have seen that the relative informality of these specifications may cause problems, the issue becomes even more serious when considering the *problems* that these solutions are meant to address. Usually the problems that various features of the solution are meant to address are not described, or only very informally. Effectively many features have no clear requirement, and the advice to someone developing a PKI reduces to “well, this is the effect extension X has on path validation, so if that is what you need, use it”.

An example is certificate policies, where their intended meaning in end-entity certificates is perhaps clear (though what happens if a certificate has multiple policies, and their policy meanings are inconsistent?), but in CA certificates their effect is so complicated that no simple requirement is ever given for them. Perhaps another example is policy qualifiers; these allow general information to annotate certificate policies, but their uses to date have been very restricted (e.g. pointers to where the policy definition may be obtained), and it is completely unclear what applications should do with them, and what would be acceptable ways for CAs to populate them (thankfully the path validation decision is not to be dependent upon policy qualifiers; the new PKIX algorithm does specify how to process qualifiers for “any-policy”, and hence be generating a “solution” based on semantics for qualifiers, which may ultimately be inappropriate).

Even when standards are correct, we can often provide an accurate and more abstract re-expression of the standard. This can be particularly useful for implementations that seek to use algorithms or data structures which differ from those used in the standards (although, of course, aiming to be behaviourally equivalent).

There exists the potential to prove certain properties about a PKI on the basis of the standard it implements. This could serve as a valuable input to the development of any future “higher-grade” PKI developments.

6 Conclusions and Future Work

We have seen how an absence of formal specification in a more intricate part of a technical standard has caused problems with implementations, and resulted in a general interpretation apparently diverging from that intended by its authors (hence leading to the need for revisions of these parts). By adding formality in the specification process, we suggest that some of these problems may be avoided.

In the specific case of path validation, we have explained how a simple model (our framework) can be used to bring order to the many checks required in path validation, and provide a basis for formalisation. We have, furthermore, explained some aspects of just such a formalisation, linking it back to statements within the standard. Thus the feasibility of such an approach to specification in this domain area is demonstrated.

In future work we would like to extend further into the areas of PKIX path validation not yet modelled, and also look at modelling and comparing some alternatives for resolving the current problems with this validation algorithm. It may be possible to extend the signature theory in Section 4.2 by adding the concept of non-deterministic parameters, so that it could accommodate DSA. We could also extend the work further to cover more of the total certificate lifecycle (e.g. registration authority functions, rekeying, and revocation). Closely related to these aspects are the underlying trust models; clearly this would relate to work by Maurer [10], Reiter and Stubblebine [16,17], and others.

Acknowledgements. We would like to thank Dr. Brendan Mahony and Dr. Jim McCarthy for helpful discussions on Isabelle theories. Thanks are also due to Peter Drewer, the Head of Trusted Computer Systems Group, who carefully read the paper and gave useful comments. Thanks also to Peter Gutmann for valuable information about the “Cryptlib” implementation. We would also like to take this opportunity to thank anonymous referees for valuable comments and suggestions.

References

1. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. In *Proceedings of the Royal Society of London* **426**, pages 233–271, 1989.
2. Cryptlib. [Online]<<http://www.datasec.co.nz>>, Technical Contact: Peter Gutmann.
3. DR 222. *Defect Report Number DR 222: Certificate Policy Mapping*. Available from <ftp://ftp.bull.com/pub/OSIdirectory/DefectResolution/DefectReports/X.509/>, June 1999.
4. W. Ford and M. Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption*. Prentice-Hall, 1997.
5. M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.
6. R. Housley, W. Ford, W. Polk, and D. Solo. RCF 2459, *Internet X.509 Public Key Infrastructure - Part I: Certificate and CRL Profile*. Internet Request for Comments 2459, January 1999.
7. C. Liu, M. A. Ozols, M. Henderson, and T. Cant. A state-based model for certificate management systems. In *Proceedings of the 3rd International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2000.
8. C. Liu, M. A. Ozols, M. Henderson, and T. Cant. Towards certificate verification in a certificate management system. In Jenny Edwards, editor, *Proceedings of the 23rd Australasian Computer Science Conference, ACSC2000*, pages 150–157, Canberra, Australia, 31 January – 3 February 2000. IEEE Computer Society.
9. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocols using CSP and FDR. In T. Margaria and B. Steffen, editors, *Tool and Algorithms for the Construction and Analysis of Systems. Second International Workshop, TACAS'96, LNCS 1055*, pages 147–166. Springer, 1996.
10. Ueli Maurer. Modeling a public-key infrastructure. In E. Bertino, H. Knurth, G. Martella, and E. Montolivo, editors, *Computer Security – ESORICS'96 (LNCS 1146)*. Springer-Verlag, 1996.

11. J. Mulvenna, L. Keys, D. Walters, S. Ganta, and S. Gupta. Characteristics and attributes that affect S/MIME product interoperability. Draft available from <http://csrc.nist.gov/pki/smime/welcome.htm>.
12. NIST. *Digital Signature Standard* (DSS). Federal Information Processing Standards Publication 186, November 1994.
13. L. C. Paulson. *ML for Working Programmer*. 2nd edition, Cambridge University Press, 1996.
14. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.
15. Lawrence C. Paulson. The Isabelle Reference Manual. University of Cambridge, Computer Laboratory, available from <http://www.in.tum.de/isabelle/dist>.
16. M. K. Reiter and S. G. Stubblebine. Path independence for authentication in large-scale systems. In *Proceedings of the 4th ACM Conference on Computer and Communication Security*, pages 57–66, April 1997.
17. M. K. Reiter and S. G. Stubblebine. Toward acceptable metrics of authentication. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1997.
18. A. W. Roscoe. Modelling and verifying key-exchange protocols using csp and fdr. In *8th Computer Security Foundation Workshop*, pages 98–107. IEEE Computer Society Press, 1995.
19. B. Schneier. *Applied Cryptography*. 2nd edition, John Wiley & Sons, Inc., 1996.
20. X.509. *Information Technology - Open systems Interconnection - The Directory: Authentication Framework*. ITU-T Recommendation X.509, June 1997.

Iterative Probabilistic Cryptanalysis of RC4 Keystream Generator

Jovan Dj. Golić

School of Electrical Engineering, University of Belgrade
Bulevar Revolucije 73, 11001 Belgrade, Yugoslavia
golic@galeb.etf.bg.ac.yu

Abstract. An iterative probabilistic method for reconstructing the initial state of RC4 keystream generator from a short segment of the keystream sequence is developed. The cryptanalytic algorithm consists of a forward and backward recursive computation of the posterior probabilities of the internal state components given a keystream segment. While maintaining the computational complexity, the new method presents a theoretical and practical improvement of a recently proposed method of a similar type, as less entries of the initial table are required to be known for the attack to be successful. If these entries have to be guessed, then the attack is expected to remain infeasible for the recommended word size of RC4.

Key words. Keystream generator, cryptanalysis, recursive probabilities, iterative algorithms, permutations.

1 Introduction

Keystream generators for practical stream cipher applications can generally be represented as autonomous finite-state machines whose initial state and possibly the next-state and output functions are secret-key-dependent. A common type of keystream generators consists of a number of possibly irregularly clocked linear feedback shift registers that are combined by a function with or without memory. Such generators are relatively well explored in the open literature. They can possess desired cryptographic properties, but under certain conditions, they may be vulnerable to various divide-and-conquer attacks in the known plaintext (or ciphertext-only) scenario (for a survey, see [7] and [4]).

A different design approach is taken in a software-oriented keystream generator [8] publicized in [9] and known as RC4. It is widely used in a number of commercial products and standards including Secure Sockets Layer standard SSL 3.0. Given a parameter n (nominally, $n = 8$), the internal state of RC4 consists of a balanced table (permutation) of 2^n different n -bit words and two pointer n -bit words which, at each time, define the positions of two words in the table to be swapped to produce the table at the next time. One of the pointers is updated by using the table content at the position defined by the other, which is itself updated in a known way by a counter. Initially, the two pointer

words are set to zero and the table content is defined by the secret key. The table thus varies slowly in time under the control of itself. At each time, the output of RC4 is an n -bit word which is taken from an appropriate position in the table. The output word is then bitwise added to the plaintext word to give the ciphertext word. The effective internal memory size in bits is large and is given as $M = \log 2^n! + 2n \approx 2^n(n - \log e) + 5n/2 + \log \sqrt{2\pi}$, where the logarithm is to the base 2.

Standard cryptographic properties are hard to prove for RC4. According to the linear model approach [2], a linear statistical weakness of the RC4 keystream is both theoretically and experimentally established in [3] and [5]. The keystream sequence length needed to detect the weakness is considerably shorter than the period and may even be realistic in high speed applications for $n \leq 8$. However, as this length is fairly large, the result is more of a theoretical than practical interest.

Two cryptanalytic algorithms for reconstructing the initial state of RC4 from a short segment of the keystream are proposed in [6]. One consists of the sequential search through the values of the internal state components (one of the pointers and appropriate entries of the table at successive times) that are consistent with the given keystream segment. Its time complexity is close to the time of searching through the square root of all possible initial states. The other consists of a recursive computation of the posterior probabilities of the internal state components given the corresponding keystream symbols. It can recover the content of the initial table provided a sufficient number of its entries are known (about 60% for $n = 8$). Its time complexity is about 2^{6n} steps each consisting of computing the product of appropriate probabilities.

The main objective of this paper is to develop an iterative probabilistic method for reconstructing the initial state of RC4 from a short segment of the keystream sequence. A careful probabilistic analysis reveals two main deficiencies of the probabilistic approach from [6]. First, the expressions [6] for the posterior probabilities are just approximations as the two so-called ‘change of state’ and ‘observation of output symbol’ effects should be considered simultaneously rather than separately. Second, apart from the forward recursion of the posterior probabilities, the backward recursion of these probabilities is also required. As a result, less entries of the initial table have to be known for the attack to recover the whole initial table. This is illustrated by experimental results obtained by computer simulations. The time complexity is about 2^{6n} steps consisting of computing the product of appropriate probabilities.

In Section 2, a more detailed description of RC4 keystream generator is presented. In Section 3, general expressions for the forward and backward recursions of the posterior probabilities are determined, while the underlying conditional probabilities are derived in the Appendix. Section 4 contains a description of the corresponding iterative cryptanalytic algorithms and Section 5 is devoted to illustrative experimental results. A summary and conclusions are given in Section 6.

2 Description of RC4

We follow the description given in [9]. RC4 is in fact a family of algorithms indexed by a parameter n , which is a positive integer typically recommended to be equal to 8. The internal state of RC4 at time t consists of a table (permutation) $S_t = (S_t[l])_{l=0}^{2^n-1}$ of 2^n different n -bit words and of two pointer n -bit words i_t and j_t . The n -bit words are treated as binary representations of integers. So, the internal memory size in bits is $M = \log 2^n! + 2n$. Let the output n -bit word of RC4 at time t be denoted by Z_t . Let initially $i_0 = j_0 = 0$. Then the next-state and output functions of RC4 are for every $t \geq 1$ defined by

$$i_t = i_{t-1} + 1 \quad (1)$$

$$j_t = j_{t-1} + S_{t-1}[i_t] \quad (2)$$

$$S_t[i_t] = S_{t-1}[j_t], \quad S_t[j_t] = S_{t-1}[i_t] \quad (3)$$

$$Z_t = S_t[S_t[i_t] + S_t[j_t]] \quad (4)$$

where all the additions are modulo 2^n . It is assumed that all the words except for the swapped ones remain the same (swapping itself is effective only if $i_t \neq j_t$). The output n -bit word sequence is $Z = (Z_t)_{t=1}^{\infty}$. The initial table S_0 is derived from the secret key string in a way which is not important for our cryptanalysis.

The known pointer sequence $(i_t)_{t=0}^{\infty}$ ensures that the next-state function is invertible (one-to-one).

3 Recursive Probabilistic Analysis

In a probabilistic model where the initial permutation S_0 is chosen randomly according to the uniform probability distribution, our objective would ideally be to determine the posterior probability $\Pr\{S_0|Z_1^T\}$, where $Z_{t_1}^{t_2} = z_{t_1}, z_{t_1+1}, \dots, z_{t_2}$ denotes a segment of length $t_2 - t_1 + 1$ of the given keystream sequence. For simplicity, we keep the same notation for random variables and their values. Thus $\Pr\{x\}$ denotes the probability $\Pr\{X = x\}$, where X is a random variable. If $2^{nT} \geq 2^n!$, that is, if $T \geq (\log 2^n!)/n$, then S_0 is expected to be uniquely determined by Z_1^T . Computing this probability corresponds to the algorithm [1] for estimating the internal states of a hidden Markov chain and is equivalent to the exhaustive search over all possible S_0 . Note that the time and space complexities of the algorithm [1] are both proportional to the cardinality of the internal state space of the underlying Markov chain.

Therefore, instead of the internal state as a whole, we consider its individual components, that is, the individual entries of the table $S_t[i]$, $0 \leq i \leq 2^n - 1$, and the pointer j_t (the pointer i_t is known). Our ultimate objective is to determine the posterior probabilities $\Pr\{S_0[i]|Z_1^T\}$, $0 \leq i \leq 2^n - 1$. This can be

achieved by considering the posterior probabilities $\Pr\{S_t[i]|Z_1^T\}$, $0 \leq i \leq 2^n - 1$, and $\Pr\{j_t|Z_1^T\}$, for $0 \leq t \leq T$. The algorithm [1] is not applicable because we deal with the internal state components individually rather than with the whole internal state.

The essence of our probabilistic method is to derive the forward recursions for the posterior probabilities $\Pr\{S_t[i]|Z_1^t\}$, $0 \leq i \leq 2^n - 1$, and $\Pr\{j_t|Z_1^t\}$, for $1 \leq t \leq T$, and the backward recursions for the posterior probabilities $\Pr\{S_t[i]|Z_{t+1}^T\}$, $0 \leq i \leq 2^n - 1$, and $\Pr\{j_t|Z_{t+1}^T\}$, for $0 \leq t \leq T - 1$. Both directions are needed since the underlying independence assumption gives rise only to approximate expressions. It is important to deal with Z_{t+1}^T rather than Z_t^T , as the backward recursions for $\Pr\{S_t[i]|Z_t^T\}$ and $\Pr\{j_t|Z_t^T\}$ are much more difficult to derive and compute. Unlike the algorithm [1] where the forward and backward recursions are essentially the same, the forward and backward recursions derived here are different, because the Markov chain property does not hold for the individual internal state components.

3.1 Forward Recursions

For the forward direction, note that the next-state and output functions can be put into the form

$$j_t = j_{t-1} + S_{t-1}[i_t], \quad i_t = i_{t-1} + 1 \quad (5)$$

$$S_t[i] = \begin{cases} S_{t-1}[j_t], & i = i_t \\ S_{t-1}[i_t], & i = j_t \\ S_{t-1}[i], & i \neq i_t, j_t \end{cases} \quad (6)$$

$$Z_t = S_t[S_{t-1}[i_t] + S_{t-1}[j_t]]. \quad (7)$$

In view of (5)-(7), it is convenient to perform the following Bayesian decomposition

$$\begin{aligned} \Pr\{S_t[i]|Z_1^t\} &= \frac{\Pr\{Z_1^{t-1}\}}{\Pr\{Z_1^t\}} \cdot \sum_{j_t, S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i]} \\ &\Pr\{S_t[i], Z_t|j_{t-1} = j_t - S_{t-1}[i_t], S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i], Z_1^{t-1}\} \cdot \\ &\Pr\{j_{t-1} = j_t - S_{t-1}[i_t], S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i]|Z_1^{t-1}\} \end{aligned} \quad (8)$$

where the summation is over all permissible pointer and entry values involved. The j_t pointer can take any possible value, whereas different entries of the table must take different values, because S_{t-1} is a permutation. Note that, for convenience, the summation is over j_t although the conditioning is on j_{t-1} .

Equation (8) gives rise to a recursion by using an approximation that the j_{t-1} pointer and different entries of S_{t-1} are independent of each other when

conditioned on Z_1^{t-1} , except that different entries must take different values. Accordingly, (8) can be approximated as

$$\begin{aligned} \Pr\{S_t[i]|Z_1^t\} &= \frac{\Pr\{Z_1^{t-1}\}}{\Pr\{Z_1^t\}} \cdot \sum_{j_t, S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i]} \\ &\Pr\{S_t[i], Z_t[j_{t-1} = j_t - S_{t-1}[i_t], S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i], Z_1^{t-1}\} \cdot \\ &\Pr\{j_{t-1} = j_t - S_{t-1}[i_t]|Z_1^{t-1}\} \cdot \Pr\{S_{t-1}[i_t]|Z_1^{t-1}\} \cdot \\ &\frac{\Pr\{S_{t-1}[j_t]|Z_1^{t-1}\}}{\Pr\{S_{t-1}[j_t] \neq S_{t-1}[i_t]|Z_1^{t-1}\}} \cdot \frac{\Pr\{S_{t-1}[i]|Z_1^{t-1}\}}{\Pr\{S_{t-1}[i] \neq S_{t-1}[i_t], S_{t-1}[j_t]|Z_1^{t-1}\}} \end{aligned} \quad (9)$$

(where formally $0/0 = 0$). It is assumed that the probability of each entry of S_{t-1} , when the values of i_t , j_t , and i are not different, is included only once.

Note that the used approximation of the joint probability of different entries of S_{t-1} , under the *independence assumption* as in [6], depends on the chosen order of these entries (here i_t , j_t , i). Interestingly, another approximation, which is simpler to compute, proves to be somewhat more successful in the experiments reported in Section 5. Namely, under the *simplified independence assumption*, the joint probability of different entries can be approximated just as the product of their individual probabilities. Thus, the last line of (9) is then replaced by

$$\Pr\{S_{t-1}[j_t]|Z_1^{t-1}\} \cdot \Pr\{S_{t-1}[i]|Z_1^{t-1}\}.$$

The multiplicative term $\Pr\{Z_1^{t-1}\}/\Pr\{Z_1^t\}$ is determined from the normalization requirement

$$\sum_{S_t[i]} \Pr\{S_t[i]|Z_1^t\} = 1. \quad (10)$$

Similarly, we can obtain a forward recursion for the j_t pointer as

$$\begin{aligned} \Pr\{j_t|Z_1^t\} &= \frac{\Pr\{Z_1^{t-1}\}}{\Pr\{Z_1^t\}} \cdot \\ &\sum_{S_{t-1}[i_t], S_{t-1}[j_t]} \Pr\{j_t, Z_t[j_{t-1} = j_t - S_{t-1}[i_t], S_{t-1}[i_t], S_{t-1}[j_t], Z_1^{t-1}\} \cdot \\ &\Pr\{j_{t-1} = j_t - S_{t-1}[i_t]|Z_1^{t-1}\} \cdot \Pr\{S_{t-1}[i_t]|Z_1^{t-1}\} \cdot \\ &\frac{\Pr\{S_{t-1}[j_t]|Z_1^{t-1}\}}{\Pr\{S_{t-1}[j_t] \neq S_{t-1}[i_t]|Z_1^{t-1}\}}. \end{aligned} \quad (11)$$

where the multiplicative term $\Pr\{Z_1^{t-1}\}/\Pr\{Z_1^t\}$ can be determined from the normalization requirement

$$\sum_{j_t} \Pr\{j_t|Z_1^t\} = 1. \quad (12)$$

Note that the conditioning in (11) is on j_{t-1} . Under the simplified independence assumption, the last line of (11) is replaced by

$$\Pr\{S_{t-1}[j_t]|Z_1^{t-1}\}.$$

The conditional probabilities in the second lines of (9) and (11), that is,

$$\Pr\{S_t[i], S_t[S_{t-1}[i_t] + S_{t-1}[j_t]] = Z_t | \\ j_{t-1} = j_t - S_{t-1}[i_t], S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i], Z_1^{t-1}\} \quad (13)$$

and

$$\Pr\{j_t, S_t[S_{t-1}[i_t] + S_{t-1}[j_t]] = Z_t | \\ j_{t-1} = j_t - S_{t-1}[i_t], S_{t-1}[i_t], S_{t-1}[j_t], Z_1^{t-1}\} \quad (14)$$

are determined in Appendix A by using (5)-(7).

3.2 Backward Recursions

As the next-state function of RC4 is one-to-one, the next-state and output functions for the backward direction in time can be written as

$$j_t = j_{t+1} - S_{t+1}[j_{t+1}], \quad i_t = i_{t+1} - 1 \quad (15)$$

$$S_t[i] = \begin{cases} S_{t+1}[j_{t+1}], & i = i_{t+1} \\ S_{t+1}[i_{t+1}], & i = j_{t+1} \\ S_{t+1}[i], & i \neq i_{t+1}, j_{t+1} \end{cases} \quad (16)$$

$$Z_{t+1} = S_{t+1}[S_{t+1}[i_{t+1}] + S_{t+1}[j_{t+1}]]. \quad (17)$$

In view of (15)-(17), under similar assumptions, we obtain the following analogs of (8)-(11), respectively:

$$\Pr\{S_t[i] | Z_{t+1}^T\} = \frac{\Pr\{Z_{t+2}^T\}}{\Pr\{Z_{t+1}^T\}} \cdot \sum_{j_{t+1}, S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], S_{t+1}[i]} \\ \Pr\{S_t[i], Z_{t+1}[j_{t+1}], S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], S_{t+1}[i], Z_{t+2}^T\} \cdot \\ \Pr\{j_{t+1}, S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], S_{t+1}[i] | Z_{t+2}^T\} \quad (18)$$

$$\Pr\{S_t[i] | Z_{t+1}^T\} = \frac{\Pr\{Z_{t+2}^T\}}{\Pr\{Z_{t+1}^T\}} \cdot \sum_{j_{t+1}, S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], S_{t+1}[i]} \\ \Pr\{S_t[i], Z_{t+1}[j_{t+1}], S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], S_{t+1}[i], Z_{t+2}^T\} \cdot \\ \Pr\{j_{t+1} | Z_{t+2}^T\} \cdot \Pr\{S_{t+1}[i_{t+1}] | Z_{t+2}^T\} \cdot \frac{\Pr\{S_{t+1}[j_{t+1}] | Z_{t+2}^T\}}{\Pr\{S_{t+1}[j_{t+1}] \neq S_{t+1}[i_{t+1}] | Z_{t+2}^T\}} \cdot \\ \frac{\Pr\{S_{t+1}[i] | Z_{t+2}^T\}}{\Pr\{S_{t+1}[i] \neq S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}] | Z_{t+2}^T\}} \quad (19)$$

$$\begin{aligned}
\Pr\{j_t|Z_{t+1}^T\} &= \frac{\Pr\{Z_{t+2}^T\}}{\Pr\{Z_{t+1}^T\}} \cdot \sum_{S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}]} \\
&\Pr\{j_t, Z_{t+1}|j_{t+1} = j_t + S_{t+1}[j_{t+1}], S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], Z_{t+2}^T\} \cdot \\
&\Pr\{j_{t+1} = j_t + S_{t+1}[j_{t+1}]|Z_{t+2}^T\} \Pr\{S_{t+1}[i_{t+1}]|Z_{t+2}^T\} \cdot \\
&\frac{\Pr\{S_{t+1}[j_{t+1}]|Z_{t+2}^T\}}{\Pr\{S_{t+1}[j_t] \neq S_{t+1}[i_{t+1}]|Z_{t+2}^T\}}. \tag{20}
\end{aligned}$$

Note that the conditioning in (20) is on j_{t+1} .

Under the simplified independence assumption, the last two lines of (19) and the last line of (20) are modified analogously. Note that the Bayesian decompositions in (18)-(20) are carefully chosen so as to make the computations as simple as possible. The analogy with (8)-(11) is considerable, but not complete.

The conditional probabilities in the second lines of (19) and (20), that is,

$$\begin{aligned}
&\Pr\{S_t[i], S_{t+1}[S_{t+1}[i_{t+1}] + S_{t+1}[j_{t+1}]] = Z_{t+1}| \\
&\quad j_{t+1}, S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], S_{t+1}[i], Z_{t+2}^T\} \tag{21}
\end{aligned}$$

and

$$\begin{aligned}
&\Pr\{j_t, S_{t+1}[S_{t+1}[i_{t+1}] + S_{t+1}[j_{t+1}]] = Z_{t+1}| \\
&\quad j_{t+1} = j_t + S_{t+1}[j_{t+1}], S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}], Z_{t+2}^T\} \tag{22}
\end{aligned}$$

are determined in Appendix B by using (15)-(17).

The multiplicative term $\Pr\{Z_{t+2}^T\}/\Pr\{Z_{t+1}^T\}$ is determined from similar normalization requirements.

4 Iterative Algorithms

The forward recursion (9) together with the conditional probability (13) determined in Appendix A can be used to compute the posterior probabilities $\Pr\{S_t[i]|Z_1^t\}$, $0 \leq S_t[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, from the previously computed posterior probabilities $\Pr\{S_{t-1}[k]|Z_1^{t-1}\}$, $0 \leq S_{t-1}[k] \leq 2^n - 1$, $0 \leq k \leq 2^n - 1$, and $\Pr\{j_{t-1}|Z_1^{t-1}\}$, $0 \leq j_{t-1} \leq 2^n - 1$. Note that the summation is effectively only over the variables j_t , $S_{t-1}[i_t]$, and $S_{t-1}[j_t]$, as $S_{t-1}[i]$ is in all the cases uniquely determined by the remaining variables and the given values of i , $S_t[i]$, i_t , and Z_t . Similarly, the forward recursion (11) together with the conditional probability (14) determined in Appendix A can be used to compute the posterior probabilities $\Pr\{j_t|Z_1^t\}$, $0 \leq j_t \leq 2^n - 1$, from the previously computed posterior probabilities $\Pr\{S_{t-1}[k]|Z_1^{t-1}\}$, $0 \leq S_{t-1}[k] \leq 2^n - 1$, $0 \leq k \leq 2^n - 1$, and $\Pr\{j_{t-1}|Z_1^{t-1}\}$, $0 \leq j_{t-1} \leq 2^n - 1$.

For each t , $1 \leq t \leq T$, the time complexities of updating the posterior probabilities for $S_t[i]$ and j_t are thus about 2^{5n} and 2^{3n} steps, respectively, each step consisting of computing the product of the involved probabilities.

- (1) We thus define the procedure FORWARD that, given the parameters 2^n , T , and Z_1^T , computes the output (terminal) probability distributions $\Pr\{S_T[i]|Z_1^T\}$, $0 \leq S_T[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, and $\Pr\{j_T|Z_1^T\}$, $0 \leq j_T \leq 2^n - 1$, from the given input (initial) probability distributions $\Pr\{S_0[i]\}$, $0 \leq S_0[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, and (fixed) $\Pr\{j_0\} = \delta_{0,j_0}$, $0 \leq j_0 \leq 2^n - 1$ (where $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$).

Analogously, the backward recursions (19) and (20) together with the corresponding conditional probabilities (21) and (22) determined in Appendix B can be used to compute the posterior probabilities $\Pr\{S_t[i]|Z_{t+1}^T\}$, $0 \leq S_t[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, and $\Pr\{j_t|Z_{t+1}^T\}$, $0 \leq j_t \leq 2^n - 1$, respectively, from the previously computed posterior probabilities $\Pr\{S_{t+1}[k]|Z_{t+2}^T\}$, $0 \leq S_{t+1}[k] \leq 2^n - 1$, $0 \leq k \leq 2^n - 1$, and $\Pr\{j_{t+1}|Z_{t+2}^T\}$, $0 \leq j_{t+1} \leq 2^n - 1$. Also, for each t , $T - 1 \geq t \geq 0$, the time complexities of updating the posterior probabilities for $S_t[i]$ and j_t are thus 2^{5n} and 2^{3n} corresponding steps, respectively.

- (2) We analogously define the procedure BACKWARD that, given the parameters 2^n , T , and Z_1^T , computes the output (initial) probability distributions $\Pr\{S_0[i]|Z_1^T\}$, $0 \leq S_0[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, from the given input (terminal) probability distributions $\Pr\{S_T[i]\}$, $0 \leq S_T[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, and $\Pr\{j_T\}$, $0 \leq j_T \leq 2^n - 1$. For control, one can also produce the probability distribution $\Pr\{j_0|Z_1^T\}$, $0 \leq j_0 \leq 2^n - 1$.
- (3) The basic iterative algorithm IA is composed of M rounds, each round consisting of the procedure FORWARD followed by the procedure BACKWARD, where the output probability distributions of FORWARD are used as the input probability distributions of BACKWARD, and where the input probability distributions of FORWARD in each round are defined as the output probability distributions of BACKWARD in the previous round. Initially, in the first round, the uniform probability distributions $\Pr\{S_0[i]\} = 2^{-n}$, $0 \leq S_0[i] \leq 2^n - 1$, $0 \leq i \leq 2^n - 1$, are used. The output probability distributions of BACKWARD in the last round represent the soft estimate of the initial table S_0 that produced the given keystream segment Z_1^T .

The experiments obtained by computer simulations show that the iterative algorithm in its basic form cannot recover the initial table S_0 . However, a modified version in which a given number of entries of S_0 , that is, $S_0[i]$, $1 \leq i \leq \tau$, are known can be successful provided that τ is sufficiently large. In this case, the input probability distributions of FORWARD are adjusted accordingly in each round. If these entries are not known, then, at least in theory, they can be guessed either by exhaustive search or by one of the methods from [6]. This, of course, is not feasible if τ is large. Note that each guess is easily checked for correctness by comparing the given keystream sequence with the output sequence produced from the reconstructed initial table.

The minimal keystream sequence length T required for a successful attack can be estimated as the minimal m such that $2^{nm} \geq 2^n!$ ($T_{\min} = 5, 11, 24, 50, 103, 211$ for $n = 3, 4, 5, 6, 7, 8$, respectively). Clearly, $T_{\min} < 2^n$. If τ entries are initially

known, then this estimate can be reduced accordingly. The total time complexity of IA is thus about 2^{6n} corresponding steps.

Instead of the independence assumption, we can also work with the simplified independence assumption. Accordingly, by using the corresponding forward and backward recursions and conditional probabilities, we analogously define the simplified procedures FORWARD' and BACKWARD' and the simplified iterative algorithm IA'. Although the total number of computational steps remains the same, the complexity of most steps is thus reduced.

5 Preliminary Experimental Results

The objective of experiments conducted by computer simulations was to verify if the iterative algorithms IA and IA' can recover the initial permutation S_0 and to investigate the features of these algorithms. In each experiment, the keystream segment Z_1^T is first generated from a pseudorandomly chosen S_0 . Then for both IA and IA', the minimal τ as well as the minimal number of rounds, M , for such τ that are required for successful recovery are found. The choice of the keystream sequence length T was also examined.

Numerous experiments were performed for $n = 3$ and $n = 4$, whereas the results for larger n require more computational power than the one available. Consequently, the following observations regarding the general features of the algorithms IA and IA' can be made.

- (1) Neither IA nor IA', in their basic form, are successful for $\tau = 0$. This is probably because the (simplified) independence assumption is only an approximation. More precisely, if the considered time t increases, then different entries of the table S_t become more dependent on each other when conditioned on Z_1^t . It is interesting to investigate if the algorithms IA and IA' or the procedures FORWARD/BACKWARD and FORWARD'/BACKWARD' can be (considerably) improved.
- (2) For most permutations S_0 , $\tau_{\min} = 1$ or 2 for $n = 3$ and $\tau_{\min} = 4$ or 5 for $n = 4$. Accordingly, for $n = 8$, we may expect that $\tau_{\min}$ is much smaller than 155 (see [6]). This remains to be verified by experiments. Theoretical analysis of $\tau_{\min}$ as a function of S_0 , given n , seems to be difficult.
- (3) In many cases (especially if $\tau_{\min} = 1$, $n = 3$, and $\tau_{\min} = 4$, $n = 4$), it was found that $M_{\min} > 1$ (e.g., $M_{\min} = 2, 3, 4, 5$). In many successful experiments, it was observed that the j_t pointer sequence was not uniquely determined in the first pass of FORWARD (unlike [6]).
- (4) In most experiments, IA and IA' were equally successful (the same $\tau_{\min}$) and in a number of cases, IA' converged faster than IA (smaller $M_{\min}$). This is somewhat surprising, because IA' is simpler than IA.
- (5) For convenience, in most experiments it was assumed that $T = 2^n$. However, it was observed that in some cases decreasing T resulted in decreasing $M_{\min}$ or even $\tau_{\min}$. This means that increasing T can sometimes result in failure, and IA' is more sensitive than IA in this respect.

- (6) In the case of success, the probabilities in both IA and IA' converge to either 0 or 1 thus yielding the correct hard estimates of S_0 . In the case of failure, the probability distributions either remain relatively flat or sometimes yield wrong hard or mixed hard/soft estimates of S_0 . Adaptation of the algorithms so as to make use of partially correct hard estimates of S_0 can be explored in future. Other ways of combining the procedures FORWARD and BACKWARD (FORWARD' and BACKWARD') may also be investigated.

6 Conclusions

An iterative probabilistic method for reconstructing the initial state of RC4 from a short segment of the keystream sequence is developed. The probabilistic approach from [6] is used as a starting point, but the conducted probabilistic analysis is more sophisticated. The method consists of a recursive computation of the posterior probabilities of the internal state components given a keystream segment. The forward and backward recursions of the posterior probabilities are determined under the so-called independence and simplified independence assumptions. The underlying conditional probabilities are presented in sufficient detail for implementation. The corresponding iterative algorithms are implemented in software. The experimental results obtained by computer simulations are used to establish a number of features of the proposed algorithms. As a number of entries of the initial table have to be guessed correctly for the algorithms to be successful, the new attack, although more effective than the one from [6], is expected to remain unpractical for the recommended word size of RC4.

Unlike [6], besides the forward recursions, our method also involves the backward recursions, and the derived expressions are more precise than the ones from [6], because the so-called 'change of state' and 'observation of output symbol' effects are considered simultaneously rather than separately. In addition, it is established that the introduced simplified independence assumption gives rise to a faster, yet equally successful algorithm. The proposed cryptanalytic algorithms have roughly the same time complexity as the corresponding algorithm [6], but are more effective in that they require less entries of the initial table to be known for the attack to recover the whole initial table. Whether the algorithms can be improved by taking into account the pairwise dependences between different entries of the table is an interesting problem for future investigations.

Appendix

A Conditional Probabilities: Forward Direction

Let $\sigma = S_{t-1}[i_t] + S_{t-1}[j_t]$. In order to compute (13), we distinguish between the following 15 cases, depending on the values of i , i_t , j_t , and σ being equal to or different from each other. We utilize (5)-(7), the independence assumption, and the permutation constraint that at any time different entries of the table must

take different values. Let P denote the conditional probability in question. All the cases are specified in sufficient detail for implementation.

Case 1 $i \neq i_t, j_t \neq i_t, i, \sigma \neq i_t, j_t, i$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t], S_{t-1}[j_t], S_{t-1}[i] | Z_1^{t-1}\}$ provided that $S_t[i] \neq Z_t, S_{t-1}[i] = S_t[i], S_{t-1}[i_t] \neq Z_t, S_t[i]$, and $S_{t-1}[j_t] \neq Z_t, S_t[i], S_{t-1}[i_t], i - S_{t-1}[i_t], i_t - S_{t-1}[i_t], j_t - S_{t-1}[i_t]$. Otherwise, $P = 0$.

Case 2 $i \neq i_t, j_t \neq i_t, i, \sigma = i$.

Then $P = 1$ provided that $S_t[i] = Z_t, S_{t-1}[i] = Z_t, S_{t-1}[j_t] = i - S_{t-1}[i_t], S_{t-1}[i_t] \neq Z_t, i - Z_t$, and $2S_{t-1}[i_t] \neq i$. Otherwise, $P = 0$.

Case 3 $i \neq i_t, j_t \neq i_t, i, \sigma = i_t$.

Then $P = 1$ provided that $S_t[i] \neq Z_t, i_t \neq 2Z_t, Z_t + S_t[i], S_{t-1}[i] = S_t[i], S_{t-1}[j_t] = Z_t$, and $S_{t-1}[i_t] = i_t - Z_t$. Otherwise, $P = 0$.

Case 4 $i \neq i_t, j_t \neq i_t, i, \sigma = j_t$.

Then $P = 1$ provided that $S_t[i] \neq Z_t, S_{t-1}[i] = S_t[i], j_t \neq 2Z_t, Z_t + S_t[i], S_{t-1}[i_t] = Z_t$, and $S_{t-1}[j_t] = j_t - Z_t$. Otherwise, $P = 0$.

Case 5 $i \neq i_t, j_t = i, \sigma \neq i_t, j_t$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t], S_{t-1}[j_t] | Z_1^{t-1}\}$ provided that $S_t[i] \neq Z_t, S_{t-1}[i] = S_{t-1}[j_t], S_{t-1}[i_t] = S_t[i]$, and $S_{t-1}[j_t] \neq Z_t, S_t[i], i - S_t[i], i_t - S_t[i]$. Otherwise, $P = 0$.

Case 6 $i \neq i_t, j_t = i, \sigma = i_t$.

Then $P = 1$ provided that $S_t[i] \neq Z_t, i_t = Z_t + S_t[i], S_{t-1}[i] = S_{t-1}[j_t], S_{t-1}[i_t] = S_t[i]$, and $S_{t-1}[j_t] = Z_t$. Otherwise, $P = 0$.

Case 7 $i \neq i_t, j_t = i, \sigma = j_t$.

Then $P = 1$ provided that $S_t[i] = Z_t, i \neq 2Z_t, S_{t-1}[i] = S_{t-1}[j_t], S_{t-1}[i_t] = Z_t$, and $S_{t-1}[j_t] = i - Z_t$. Otherwise, $P = 0$.

Case 8 $i \neq i_t, j_t = i_t, \sigma \neq i_t, i_t$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t], S_{t-1}[i] | Z_1^{t-1}\}$ provided that $S_t[i] \neq Z_t, S_{t-1}[i] = S_t[i], S_{t-1}[i_t] \neq Z_t, S_t[i], 2S_{t-1}[i_t] \neq i, i_t$, and $S_{t-1}[j_t] = S_{t-1}[i_t]$. Otherwise, $P = 0$.

Case 9 $i \neq i_t, j_t = i_t, \sigma = i$.

Then $P = 1$ provided that $S_t[i] = Z_t, S_{t-1}[i] = Z_t, 2S_{t-1}[i_t] = i, S_{t-1}[i_t] \neq Z_t$, and $S_{t-1}[j_t] = S_{t-1}[i_t]$. Otherwise, $P = 0$.

Case 10 $i \neq i_t, j_t = i_t, \sigma = i_t$.

Then $P = 1$ provided that $S_t[i] \neq Z_t, i_t = 2Z_t, S_{t-1}[i] = S_t[i]$, and $S_{t-1}[j_t] = S_{t-1}[i_t] = Z_t$. Otherwise, $P = 0$.

Case 11 $i = i_t, j_t \neq i_t, \sigma \neq i_t, j_t$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t], S_{t-1}[j_t] | Z_1^{t-1}\}$ provided that $S_t[i] \neq Z_t, S_{t-1}[i] = S_{t-1}[i_t], S_{t-1}[j_t] = S_t[i]$, and $S_{t-1}[i_t] \neq Z_t, S_t[i], i_t - S_t[i], j_t - S_t[i]$. Otherwise, $P = 0$.

Case 12 $i = i_t, j_t \neq i_t, \sigma = i_t$.

Then $P = 1$ provided that $S_t[i] = Z_t, i_t \neq 2Z_t, S_{t-1}[i] = S_{t-1}[i_t] = i_t - Z_t$, and $S_{t-1}[j_t] = Z_t$. Otherwise, $P = 0$.

Case 13 $i = i_t, j_t \neq i_t, \sigma = j_t$.

Then $P = 1$ provided that $S_t[i] \neq Z_t, i_t \neq Z_t + S_t[i] = j_t, S_{t-1}[i] = S_{t-1}[i_t] = Z_t$, and $S_{t-1}[j_t] = S_t[i]$. Otherwise, $P = 0$.

Case 14 $i = i_t = j_t$, $\sigma \neq i_t$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t] | Z_1^{t-1}\}$ provided that $S_t[i] \neq Z_t$, $i_t \neq 2S_t[i]$, $S_{t-1}[i] = S_t[i]$, and $S_{t-1}[j_t] = S_{t-1}[i_t] = S_t[i]$. Otherwise, $P = 0$.

Case 15 $i = i_t = j_t$, $\sigma = i_t$.

Then $P = 1$ provided that $S_t[i] = Z_t$, $i_t = 2Z_t$, and $S_{t-1}[i] = S_{t-1}[i_t] = S_{t-1}[j_t] = Z_t$. Otherwise, $P = 0$.

Similarly, in order to compute (14), we distinguish between the following 5 cases, depending on the values of i_t , j_t , and σ being equal to or different from each other.

Case 16 $j_t \neq i_t$, $\sigma \neq i_t, j_t$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t], S_{t-1}[j_t] | Z_1^{t-1}\}$ provided that $S_{t-1}[i_t] \neq Z_t$ and $S_{t-1}[j_t] \neq Z_t$, $S_{t-1}[i_t] = i_t - S_{t-1}[i_t]$, $j_t - S_{t-1}[i_t]$. Otherwise, $P = 0$.

Case 17 $j_t \neq i_t$, $\sigma = i_t$.

Then $P = 1$ provided that $i_t \neq 2Z_t$, $S_{t-1}[j_t] = Z_t$, and $S_{t-1}[i_t] = i_t - Z_t$. Otherwise, $P = 0$.

Case 18 $j_t \neq i_t$, $\sigma = j_t$.

Then $P = 1$ provided that $j_t \neq 2Z_t$, $S_{t-1}[i_t] = Z_t$, and $S_{t-1}[j_t] = j_t - Z_t$. Otherwise, $P = 0$.

Case 19 $j_t = i_t$, $\sigma \neq i_t$.

Then $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\} / \Pr\{S_{t-1}[\sigma] \neq S_{t-1}[i_t] | Z_1^{t-1}\}$ provided that $S_{t-1}[j_t] = S_{t-1}[i_t]$, $S_{t-1}[i_t] \neq Z_t$, and $2S_{t-1}[i_t] \neq i_t$. Otherwise, $P = 0$.

Case 20 $j_t = i_t$, $\sigma = i_t$.

Then $P = 1$ provided that $i_t = 2Z_t$ and $S_{t-1}[j_t] = S_{t-1}[i_t] = Z_t$. Otherwise, $P = 0$.

Under the simplified independence assumption, the conditional probability in Cases 1, 5, 8, 11, 14, 16, and 19 is given as $P = \Pr\{S_{t-1}[\sigma] = Z_t | Z_1^{t-1}\}$.

B Conditional Probabilities: Backward Direction

Let $\sigma = S_{t+1}[i_{t+1}] + S_{t+1}[j_{t+1}]$. In order to compute (21), we distinguish between the 15 cases, depending on the values of i , i_{t+1} , j_{t+1} , and σ being equal to or different from each other. We utilize (15)-(17), the independence assumption or the simplified independence assumption, and the permutation constraint that at any time different entries of the table must take different values. The cases can be obtained from the ones for the forward direction by using the substitution $i_t \rightarrow i_{t+1}$, $j_t \rightarrow j_{t+1}$, $Z_t \rightarrow Z_{t+1}$, $S_{t-1} \rightarrow S_{t+1}$, and $Z_1^{t-1} \rightarrow Z_{t+2}^T$, except that due to the fact that Z_{t+1} is derived from S_{t+1} rather than S_t , the roles of $S_{t+1}[i_{t+1}]$ and $S_{t+1}[j_{t+1}]$ are interchanged when $\sigma = i_{t+1}$ or $\sigma = j_{t+1}$. Consequently, Cases 3', 4', 6', 7', 12', and 13' are given below, whereas the other cases are directly obtained by the substitution described above. Let P denote the conditional probability in question.

Case 3' $i \neq i_{t+1}, j_{t+1} \neq i_{t+1}, \sigma = i_{t+1}$.

Then $P = 1$ provided that $S_t[i] \neq Z_{t+1}, i_{t+1} \neq 2Z_{t+1}, Z_{t+1} + S_t[i], S_{t+1}[i] = S_t[i], S_{t+1}[i_{t+1}] = Z_{t+1}$, and $S_{t+1}[j_{t+1}] = i_{t+1} - Z_{t+1}$. Otherwise, $P = 0$.

Case 4' $i \neq i_{t+1}, j_{t+1} \neq i_{t+1}, \sigma = j_{t+1}$.

Then $P = 1$ provided that $S_t[i] \neq Z_{t+1}, S_{t+1}[i] = S_t[i], j_{t+1} \neq 2Z_{t+1}, Z_{t+1} + S_t[i], S_{t+1}[j_{t+1}] = Z_{t+1}$, and $S_{t+1}[i_{t+1}] = j_{t+1} - Z_{t+1}$. Otherwise, $P = 0$.

Case 6' $i \neq i_{t+1}, j_{t+1} = i, \sigma = i_{t+1}$.

Then $P = 1$ provided that $S_t[i] = Z_{t+1}, i_{t+1} \neq 2Z_{t+1}, S_{t+1}[i] = S_{t+1}[j_{t+1}], S_{t+1}[i_{t+1}] = Z_{t+1}$, and $S_{t+1}[j_{t+1}] = i_{t+1} - Z_{t+1}$. Otherwise, $P = 0$.

Case 7' $i \neq i_{t+1}, j_{t+1} = i, \sigma = j_{t+1}$.

Then $P = 1$ provided that $S_t[i] \neq Z_{t+1}, i = Z_{t+1} + S_t[i], S_{t+1}[i] = S_{t+1}[j_{t+1}], S_{t+1}[i_{t+1}] = S_t[i]$, and $S_{t+1}[j_{t+1}] = Z_{t+1}$. Otherwise, $P = 0$.

Case 12' $i = i_{t+1}, j_{t+1} \neq i_{t+1}, \sigma = i_{t+1}$.

Then $P = 1$ provided that $S_t[i] \neq Z_{t+1}, i_{t+1} = Z_{t+1} + S_t[i], S_{t+1}[i] = S_{t+1}[i_{t+1}] = Z_{t+1}$, and $S_{t+1}[j_{t+1}] = S_t[i]$. Otherwise, $P = 0$.

Case 13' $i = i_{t+1}, j_{t+1} \neq i_{t+1}, \sigma = j_{t+1}$.

Then $P = 1$ provided that $S_t[i] = Z_{t+1}, j_{t+1} \neq 2Z_{t+1}, S_{t+1}[i] = S_{t+1}[i_{t+1}] = j_{t+1} - Z_{t+1}$, and $S_{t+1}[j_{t+1}] = Z_{t+1}$. Otherwise, $P = 0$.

Similarly, in order to compute (22), we distinguish between the following 5 cases, depending on the values of i_{t+1}, j_{t+1} , and σ being equal to or different from each other. The cases are different from the ones for the forward direction because, unlike $j_t, j_{t+1} = j_t + S_{t+1}[j_{t+1}]$ is variable rather than given. Under the simplified independence assumption, the conditional probability in Cases 16' and 19' is given as $P = \Pr\{S_{t+1}[\sigma] = Z_{t+1} | Z_{t+2}^T\}$.

Case 16' $j_{t+1} \neq i_{t+1}, \sigma \neq i_{t+1}, j_{t+1}$.

Then $P = \Pr\{S_{t+1}[\sigma] = Z_{t+1} | Z_{t+2}^T\} / \Pr\{S_{t+1}[\sigma] \neq S_{t+1}[i_{t+1}], S_{t+1}[j_{t+1}] | Z_{t+2}^T\}$ provided that $S_{t+1}[i_{t+1}] \neq Z_{t+1}, j_t$ and $S_{t+1}[j_{t+1}] \neq Z_{t+1}, S_{t+1}[i_{t+1}], i_{t+1} - S_{t+1}[i_{t+1}], i_{t+1} - j_t$. Otherwise, $P = 0$.

Case 17' $j_{t+1} \neq i_{t+1}, \sigma = i_{t+1}$.

Then $P = 1$ provided that $i_{t+1} \neq 2Z_{t+1}, j_t \neq Z_{t+1}, S_{t+1}[i_{t+1}] = Z_{t+1}$, and $S_{t+1}[j_{t+1}] = i_{t+1} - Z_{t+1}$. Otherwise, $P = 0$.

Case 18' $j_{t+1} \neq i_{t+1}, \sigma = j_{t+1}$.

Then $P = 1$ provided that $j_t \neq Z_{t+1}, i_{t+1} - Z_{t+1}, S_{t+1}[j_{t+1}] = Z_{t+1}$, and $S_{t+1}[i_{t+1}] = j_t$. Otherwise, $P = 0$.

Case 19' $j_{t+1} = i_{t+1}, \sigma \neq i_{t+1}$.

Then $P = \Pr\{S_{t+1}[\sigma] = Z_{t+1} | Z_{t+2}^T\} / \Pr\{S_{t+1}[\sigma] \neq S_{t+1}[i_{t+1}] | Z_{t+2}^T\}$ provided that $i_{t+1} \neq 2j_t, j_t \neq i_{t+1} - Z_{t+1}$, and $S_{t+1}[j_{t+1}] = S_{t+1}[i_{t+1}] = i_{t+1} - j_t$. Otherwise, $P = 0$.

Case 20' $j_{t+1} = i_{t+1}, \sigma = i_{t+1}$.

Then $P = 1$ provided that $i_{t+1} = 2Z_{t+1}, j_t = Z_{t+1}$, and $S_{t+1}[j_{t+1}] = S_{t+1}[i_{t+1}] = Z_{t+1}$. Otherwise, $P = 0$.

Acknowledgement. The author is grateful to Dejan Tošić for help provided with computer simulations.

References

1. L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, Mar. 1974.
2. J. Dj. Golić, "Linear models for keystream generators," *IEEE Trans. Computers*, vol. C-45, pp. 41-49, Jan. 1996.
3. J. Dj. Golić, "Linear statistical weakness of alleged RC4 keystream generator," *Advances in Cryptology - EUROCRYPT '97, Lecture Notes in Computer Science*, vol. 1233, W. Fumy ed., Springer-Verlag, pp. 226-238, 1997.
4. J. Dj. Golić, "Recent advances in stream cipher cryptanalysis," *Publications de l'Institut Mathématique*, vol. 64/78, pp. 183-204, 1998.
5. J. Dj. Golić, "Linear models for a time-variant permutation generator," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 2374-2382, Nov. 1999.
6. L. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege, "Analysis methods for (alleged) RC4," *Advances in Cryptology - ASIACRYPT '98, Lecture Notes in Computer Science*, vol. 1514, K. Ohta and D. Pei eds., Springer-Verlag, pp. 327-341, 1998.
7. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
8. R. L. Rivest, "The RC4 encryption algorithm," RSA Data Security, Inc., Mar. 1992.
9. B. Schneier, *Applied Cryptography*. New-York: Wiley, 1996.

Security Weaknesses in a Randomized Stream Cipher

Niels Ferguson¹, Bruce Schneier¹, and David Wagner²

¹ Counterpane Systems; 101 E Minnehaha Parkway,
Minneapolis, MN 55419, USA;
{niels, schneier}@counterpane.com

² University of California Berkeley, Soda Hall,
Berkeley, CA 94720, USA;
daw@cs.berkeley.edu

Abstract. TriStrata appears to have implemented a variation of Maurer’s randomised cipher. We define a variation of Maurer’s cipher that appears to be similar to the TriStrata version, and show several crypt-analytical attacks against our variant.

1 Introduction

In 1990, Maurer introduced an information-theoretic provably-secure randomised cipher [Mau90]. Using a large pool of public random data, he shows how to use a key made up of multiple pointers into the pool to create a series of random streams to XOR into the plaintext. Someone with the same key can XOR the same random streams into the ciphertext to recover the plaintext; an analyst without the key must brute-force search the set of possible pointers to recover the plaintext.

Recently, TriStrata Corporation [Tri98,Bec98] implemented a complexity-theoretic variation of that technique as a proprietary encryption algorithm. In this paper we first present what we infer to be the exact variation used by TriStrata, and take some initial steps in cryptanalysing the cipher.

We should point out that our analysis does *not* apply to the original cipher proposed by Maurer, where the pool is chosen to be large enough that it is infeasible to read the entire contents of the pool. Unfortunately, Maurer’s cipher is not very practical—in his paper, he proposed digitizing the surface of the moon as one means of getting enough public randomness to make the cipher work—and hence not suited to present-technology implementation. We attack a simplified version, one which is more practical.

2 The TriStrata Cipher

The TriStrata Cipher is a key stream cipher with a two-part key. The first part of the key is a 1 Mbyte (2^{20} bytes) block of random data which we call the pool. For practical reasons the pool is not changed very often. In the TriStrata system,

it appears to be fixed for a long time. Additionally, the same pool is used by all clients in the system.

The second part of the key consists of a number of pointers that point to a byte in the pool. Although our information is sketchy, we believe there are 5 pointers. Each pointer is 20 bits long, so the pointers together make up a 100-bit key.¹

To generate a byte of the key stream, the five bytes that the pointers point to are XORed together. Each of the pointers is then advanced by one byte position. If all pointers were to wrap around at the end of the pool, the resulting key stream would have a period of only 2^{20} bytes. We assume that the first pointer wraps at the end of the pool, the second pointer wraps around one byte from the end, etc. We can thus represent the keystream generation algorithm as follows:

$$k_i = \bigoplus_{j=1}^5 \text{pool}[(t_j + i) \bmod (2^{20} - i + 1)]$$

where t_j is the starting position of the j 'th pointer and k_i is the i 'th byte of the key stream. The i 'th byte of the ciphertext is formed by XORing the i 'th byte of the plaintext and k_i . This is one possible generalisation of the Morehouse variation of the Vernam cipher [Kah68].

There are of course many other possible variations. The number of pointers can be changed, as can the update rule for the pointers (increment one is the simplest update rule). In Maurer's cipher, each pointer cycles through its own unique subset of the pool. These variations do not affect the spirit of our analysis, and many of our attacks will work against such variations.

In Sections 3–5 we will discuss various attacks on this cipher. Most of our attacks are known-plaintext attacks, which corresponds to a known-keystream attack. Some of the attacks are of theoretical interest only (as certification attacks), but several are efficient enough to be of practical concern. See Table 1 for a summary of cryptanalytic results.

3 Finding the Pointers from a Known Pool

In many situations the pool does not change very often, and is not really a secret. For example, in the TriStrata system the same pool is used for all clients in the system, so it is plausible to assume that the pool is known to the attacker.² Alternatively, we might be able to recover the pool using the techniques to be described later in Section 4.

3.1 Exhaustive Search

Given 16 bytes of the key stream, we can search for a set of t_j values that would produce this key stream. It is very unlikely that one of the pointers performed a

¹ In comparison: Maurer gave an example that used 50 pointers and a pool of 2^{72} bits.

² Note that in Maurer's original system, this pool is assumed to be public information.

Table 1. A summary of our attacks on the Maurer-like stream cipher.

Attack type	Time	Space	Data	Attack model
Exhaustive search	2^{93}	—	16 bytes	Known-pool
Meet-in-the-middle	2^{57}	2^{39}	16 bytes	Known-pool
Improved MITM	2^{57}	2^{23}	16 bytes	Known-pool
Weak keys	2^{39}	2^{39}	2^{17} keys	Known-pool
Linear algebra	$\leq 2^{60}$	$\leq 2^{40}$	2^{20} bytes	Known-pointers
Exhaustive search	$\leq 2^{160}$	$\leq 2^{40}$	2^{20} bytes	Nothing known
Vigenere analysis	$\leq 2^{60}$	$\leq 2^{40}$	$2^{22.3}$ bytes	Nothing known
Diff. related-key	$\leq 2^{60}$	$\leq 2^{40}$	2^{21} bytes	Related-key; nothing known
Diff. related-key	$5 \cdot 2^{20}$	—	80 bytes	5 related keys; known-pool
Diff. fault	2^{39}	2^{19}	16 bytes	Fault attack; known-pool

wrap-around within these 16 bytes, so we assume that this did not happen. We are thus looking for 5 sub-ranges in the pool of 16 bytes each that when XORed together results in the key stream.

If all t_j values are distinct, then we have to try $\binom{2^{20}}{5} \approx 2^{93}$ different sets of values. If two t_j values are the same their contributions to the key stream cancel out, and we look for a set of 3 pointers. The extreme case is when there are two pairs of t_j values that are identical, in which case the key stream is just a sub-range of the pool. These special cases do not contribute significantly to the complexity of the attack.

If one of the pointers did wrap around, the attack fails. We can repeat the attack on the next 16 bytes of the key stream, which will most likely succeed.

We conclude that an exhaustive search over the pointer values for a known pool has a complexity of 2^{93} steps. Simple optimisations can make each of these steps extremely efficient.

3.2 Meet-in-the-Middle on the Pointers

We can improve this if we use a meet-in-the-middle attack on the pointer space. We generate all pairs (t_1, t_2) and consider the key stream contributions made by these pointers. As (t_1, t_2) produce the same result as (t_2, t_1) we can restrict ourselves to the 2^{39} cases where $t_1 < t_2$. We store these pairs in a list, and sort them lexicographically by the key stream that they generate. We have 2^{39} pairs of 5 bytes each, which requires 2.5 Terabytes of memory. This list is computed once for a given pool and then stored. The complexity of this phase is 2^{39} steps.

We then try all possible values for t_3, t_4 , and t_5 in a second phase. There are $\binom{2^{20}}{3} \approx 2^{57}$ different values (again taking advantage of the fact that swapping two t values gives the same key stream). From these three t values and the known key stream, we compute the key stream contribution of the first two pointers. We now look in the sorted list to see if there is a pair of values for (t_1, t_2) that

generates the required key stream. Each such search in the sorted list may be performed quickly using binary search or hashing.

The complexity of the second phase of the attack is about 2^{57} steps, where each step requires a lookup in a sorted list of 2^{39} elements.

We can make this attack practical by a simple divide-and-conquer technique. We split the work into 2^{16} tasks, each identified by a two-byte string σ . For each task, we first generate all pairs (t_1, t_2) as above whose contribution to the key stream start with σ . There are about 2^{23} elements in this list. If we pre-compute a table that given a two-byte string points to all places in the pool where this two-byte string occurs, then we can construct this list in about 2^{23} steps. The entire list requires about 40 Mbytes of memory, well within the range of even a standard desktop PC.

The second phase is to try all possible values for t_3 , t_4 , and t_5 with the restriction that their contribution XORed with the key stream starts with σ . We try all possible values for t_3 and t_4 , compute the desired first two bytes of the key stream contribution of t_5 and use the same pre-computed table to find these values. We then search for a suitable pair (t_1, t_2) in our list for which the rest of the key stream also matches.

For any single task σ , the first phase has a complexity of 2^{23} steps, and the second phase has a complexity of about 2^{41} steps if we use some simple ordering requirements to avoid equivalent pointer values. As we have to perform a total of 2^{16} tasks, the overall complexity of our attack is still $2^{16} \times (2^{23} + 2^{41}) \approx 2^{57}$ steps. However, this improved version has a very reasonable memory requirement, and can be spread out over many machines. This attack should certainly be considered feasible for any reasonably sized organisation.

3.3 Weak Keys

If the pointer values are generated randomly, then about one in every 2^{17} keys has at least two pointers with the same starting value. The contributions of these pointers cancel, and we are left with at most three relevant pointers. Using the techniques described here, these pointers can be found with very little work. Thus about one in every 2^{17} keys is a weak key.

With sufficient cryptanalytic targets, we could expect to break our first key after about 2^{39} steps. Build a table of 2^{39} elements by enumerating all possibilities for the first two pointers. Next, for each cryptanalytic target, check whether it forms a weak key by guessing the third pointer and doing a table lookup. This takes 2^{20} work to see if a key is weak; after 2^{17} such attempts, we expect to find a single weak key which we can break. Thus, the total complexity is 2^{39} work and 2^{39} space, assuming at least 2^{17} different keystream segments are available for analysis.

4 Finding the Pool

The attacks in Section 3, above, assumed that the pool was known to the adversary. However, as we now show, even when the pool is unknown, it may be possible to recover the pool using more sophisticated techniques, and then the techniques of Section 3 will apply.

4.1 The Known-Pointers Case: Linear Algebra

Let us assume that we know the t_j values for a given encryption. Then each known byte of the key stream gives us a linear equation in 5 values of the pool. After one million key stream bytes we expect to have a non-singular set of equations and can solve for the bytes in the pool.

If we do not use the correct values for t_j , then we can expect the set of equations to be contradictory very soon after the first million key stream bytes. This allows us to detect whether a set of values for t_j is correct.

If we are just interested in detecting whether a set of t_j values is correct, we can perform this analysis for some subset of the bits in a byte. For example, it could be done for the least significant bit of each pool byte. This optimisation requires less memory and might be faster. We have not investigated the exact performance tradeoffs of this attack in detail. It might be possible to take advantage of the highly structured form of the equations instead of using an algorithm for general linear equations.

4.2 When Pointers are Unknown: Exhaustive Search

We can use the result of Section 4.1 to mount an exhaustive search attack. For each possible set of t_j values, perform the attack of Section 4.1. If the set of equations is contradictory, the values for t_j are incorrect. If a consistent set of pool bytes is found, the full key has been recovered. This attack requires about 2^{100} steps, where each step consists of setting up just over a million equations and checking for a contradiction.

4.3 Unknown Pointers: A Vigenere Analysis

When the t_j values are unknown, we can still recover the entire pool from about $5 \cdot 2^{20}$ bytes of known keystream by treating the cipher as a multiple-loop Vigenere cipher. We let

$$k_{i,j} = \text{pool}[(t_j + i) \bmod (2^{20} - i + 1)]$$

so that $k_i = k_{i,1} \oplus \dots \oplus k_{i,5}$. Each $k_{i,j}$ takes the form of a Vigenere cipher, and their XOR is a five-loop Vigenere cipher. Here we shall ignore the fact that the five streams are related, and simply treat them as independent values.

Such ciphers can be readily cryptanalyzed by standard methods [Sin68, Tuc70], such as the application of linear algebra. In this way, with about $5 \cdot 2^{20}$ bytes of known keystream, we can completely recover the five streams $k_{i,1}, \dots, k_{i,5}$. Once the $k_{i,j}$ are known, the initial pointer values t_j and the contents of the pool can be readily obtained by inspection of the $k_{i,j}$.

The discussion in Section 2 suggested using a different modulus for each pointer, because otherwise the keystream k_i would have a relatively short period of only 2^{20} bytes. This attack shows that using a different modulus for each pointer does not extend the security of the cipher very much: not more than by a factor of five, in any case. Also, we note that this attack does not work

against the variant where each stream has the same period, because we will not be able to obtain the required quantity of known keystream material. These observations suggest that it might not make much difference whether we use the same modulus for the pointers or not.

5 Differential Fault and Related-Key Attacks

We next show that, when a few bits from the pool can be corrupted, improved attacks are available. We first discuss how to mount such attacks when the adversary can make related-key queries, and then we illustrate how random bit errors can enable a differential fault attack on the cryptosystem.

5.1 Differential Related-Key Attack on the Pool

Let us assume that the attacker can force a low-weight change in the pool. This can either be done by manipulating the distribution protocol, tampering with the stored pool, or waiting for a random bit error to occur.

In this case, the attacker can determine the distances between the different pointers by observing where the erroneous key stream deviates from the proper key stream. If two Megabytes of key stream is available, this reveals the distance between the pointers. If the attacker knows where the error in the pool occurred, then the pointer values are revealed and the pool can be reconstructed using the attacks in Section 4. If the attacker does not know this, then he has to guess the position and has to perform some attack from Section 4 at most 2^{20} times.

5.2 Differential Related-Key Attack on the Pointers

If the pool is known, a differential attack on a pointer will reveal the position of the pointer. Let us assume the attacker flips the least significant bit of pointer 1. The difference between the modified key stream and the original key stream uniquely determines the value of pointer 1. This attack can be repeated for each of the pointers, resulting in an attack that requires 5 related key queries and a complexity of $5 \cdot 2^{20}$.

The attack can be generalized to use fewer related key queries; see the next subsection for an example. -middle attack on the three-pointer cipher, as described in Sections 3.2 and 3.3; this latter computation will require 2^{39} work and 2^{20} space. In sum, the generalized attack requires one related-key query, a few bytes of known text, 2^{39} work, and 2^{20} space.

5.3 Random Error Attack

We can extend this to the case where the attacker suspects there is a random bit error in the pointers. As before, the attacker can determine one of the pointers from the difference in the key stream. The remaining 4 pointers can be found with a meet-in-the-middle attack like the one in Section 3.2 with a complexity of 2^{39} . This is a kind of differential fault analysis, where a low-weight difference in the key input yields information that helps the attacker significantly.

6 Security Improvements

The obvious way to improve the security is to increase the number of pointers. This will make many of the attacks harder to implement in practice, but also results in a slower cipher. A rough estimate shows that a version with the same size of pool but 14 pointers can be attacked in 2^{128} steps using the meet-in-the-middle attack of Section 3.2. The resulting cipher would be slower by a factor of 3 or so, and still be vulnerable to the attack in Section 4.3.

7 Uses of Maurer-Type Ciphers

Simplified variants of Maurer’s cipher might be useful for very specific applications. For example, let us suppose we need to encrypt a very long stream of data very rapidly. We generate a pool and a set of starting pointers using a cryptographically strong pseudo-random generator, and use the type of cipher we described here to encrypt the bulk data. Using 4, 8, or even 16-byte words instead of bytes would increase the efficiency of the algorithm even further. Such a construction would have to be carefully crafted and analysed. This deserves a lot of further study, and until that has been done this type of cipher should not be used.

The cipher we analyzed is similar (in some ways) to other ciphers published elsewhere. For instance, Chameleon [AM97] uses four pointers into a pool of 2^{16} 64-bit words; however, the primary difference is that Chameleon drives the pointers with another (slower) stream cipher, rather than generating them by incrementing. In fact, in Chameleon the random bit error property of Section 5.1 is a feature, not a bug: it is exploited to detect traitors. Another paper [AVV95] describes a “provably secure” stream cipher which uses a Maurer-like cipher as an internal component; there, they use eight pointers into a pool of 2^8 512-bit words, and again, the pointers are updated as in Chameleon rather than by incrementing. We leave it as an open question to extend our analysis to these cases.

8 Conclusion

Simplified Maurer-like ciphers are not always secure. We have shown that many variations can be broken far more effectively than an exhaustive search of the key space. Several of our attacks are quite practical.

Acknowledgements. We are grateful to David Bernier and to the anonymous reviewers for their helpful comments.

References

- AM97. Ross Anderson and Charalampos Maniavas, “Chameleon—A New Kind of Stream Cipher,” *FSE’97*, Springer, 1997.

- AVV95. W. Aiello, S. Venkatesan, and R. Venkatesan, "Design of practical and provably good random number generators," *SODA'95*, ACM, 1995, pp. 1–9.
- Bec98. Dan Beckman, "TriStrata: A Giant Step In Enterprise Security," *Network Computing*, 15 September 1998.
- Kah68. David Kahn, *The Codebreakers*, 1968.
- Mau90. Ueli M. Maurer, "A Provably-Secure Strongly-Randomized Cipher," *Advances in Cryptology - Eurocrypt '90*, Springer-Verlag, 1990, pp 361–373.
- Sin68. A. Sinkov, *Elementary Cryptanalysis, A Mathematical Approach*. New York: Random House, 1968.
- Tri98. TriStrata webpage, <http://www.tristrata.com>.
- Tuc70. B. Tuckerman, "A study of the Vigenere-Vernam single and multiple loop enciphering systems," IBM Research Report RC2879, 14 May 1970, Yorktown Heights NY.

Two-Stage Optimisation in the Design of Boolean Functions

John A Clark and Jeremy L Jacob

Department of Computer Science
University of York, England
{jac, jeremy}@cs.york.ac.uk

Abstract. This paper shows how suitable choice of cost function can significantly affect the power of optimisation methods for the synthesising of Boolean functions. In particular we show how simulated annealing, coupled with a new cost function motivated by Parseval's Theorem, can be used to drive the search into areas of design from which traditional techniques, such as hill-climbing, can find then find excellent solutions.

1 Introduction

Cryptography needs ways to find good Boolean functions so that ciphers can resist various forms of cryptanalytical attack (particularly linear cryptanalysis and differential cryptanalysis). The main properties required are high non-linearity and low autocorrelation [9]. Recent work [7] has investigated and compared the use of random search, hill-climbing, genetic algorithms and a hybrid approach for the derivation of Boolean functions with high non-linearity. Other work [9] has investigated the use of enhanced hill-climbing methods to derive *balanced* Boolean functions with high non-linearity and low autocorrelation.

In this paper we investigate the use of another heuristic technique, namely *simulated annealing* [4] (based on the annealing process for metals). The technique has been used by other researchers to break simple substitution and transposition ciphers [1,2] and to cryptanalyse systems based on the NP-hardness of discovering a trapdoor secret, e.g. attacks on the Permuted Perceptron Problem (PPP)[5]. In optimisation-based methods the cost (or fitness) function plays a crucial role. For highly non-linear Boolean function design extant optimisation techniques methods seek to maximise the non-linearity directly. In this paper we introduce a new cost function (motivated by Parseval's Theorem) that enables the search to reach areas of the design space from which hill-climbing techniques can be used more effectively. Using this new and simple cost function we are able to converge on areas of the solution space with high non-linearity **and** low autocorrelation.

The paper is structured as follows. In Section 2 we recap on basic Boolean function terminology. In Section 3 we outline the notion of local search, describe the simulated annealing algorithm, discuss the cost functions currently used and provide a template for enhanced cost functions. Section 4 shows how using a

two-stage strategy using our new cost function at the first stage provides better results than direct maximisation of non-linearity alone. It provides comparisons with functions derived from random generation followed by hill-climbing (which is far more effective than random generation alone as shown in [8]). In Section 4.2 we show how the basic technique can be significantly improved by suitable parameter selection.

2 Boolean Functions

This section summarises the basic cryptographic definitions needed. We shall denote the binary truth table of a Boolean function by $f : \mathcal{Z}_2^n \rightarrow \mathcal{Z}_2$ mapping each combination of n binary variables to some binary value. If the number of combinations mapping to 0 is the same as the number mapping to 1 then the function is said to be *balanced*. The *polarity truth table* is a particularly useful representation for our purposes. It is defined by $\hat{f}(x) = (-1)^{f(x)}$. Two functions f and g are said to be *uncorrelated* when $\sum_{x \in \mathcal{Z}_2^n} \hat{f}(x)\hat{g}(x) = 0$. If so, if you approximate f by using g you will be right half the time and wrong half the time. An area of particular importance for cryptanalysts is the ability to approximate a function f by a simple linear function. One of the cryptosystem designer's tasks is to make such approximation as difficult as possible (by making the function f suitably *non-linear*). We shall make use of the following terms:

Linear Boolean Function. A linear Boolean function, selected by $\omega \in \mathcal{Z}_2^n$, is denoted by $L_\omega(x) = \omega_1 x_1 \oplus \omega_2 x_2 \cdots \oplus \omega_n x_n$

Affine Function. The set of affine functions is the set of linear functions and their complements $A_{\omega,c}(x) = L_\omega(x) \oplus c$.

Walsh Hadamard Transform. For a Boolean function f the Walsh-Hadamard Transform $\hat{F}$ is defined by $\hat{F}(\omega) = \sum_{x \in \mathcal{Z}_2^n} \hat{f}(x)\hat{L}_\omega(x)$. We denote the maximum absolute value taken by the transform by $WH_{max}(f) = \max_{\omega \in \mathcal{Z}_2^n} |\hat{F}(\omega)|$. It is related to the non-linearity of f .

Non-linearity of f . The non-linearity N_f of a Boolean function f is the minimum distance to any affine function. It is given by $N_f = \frac{1}{2}(2^n - WH_{max}(f))$

Parseval's Theorem. This states that $\sum_{\omega \in \mathcal{Z}_2^n} (\hat{F}(\omega))^2 = 2^{2n}$. A consequence of this result is that $WH_{max}(f) \geq 2^{\frac{n}{2}}$. This motivates our new cost function.

Autocorrelation. The autocorrelation AC_f of a Boolean function f is given by $AC_f = \max_s \left| \sum_x \hat{f}(x)\hat{f}(x \oplus s) \right|$ Here x and s range over $\mathcal{Z}_2^n$ and $x \oplus s$ denotes bitwise XOR (and so produces a result in $\mathcal{Z}_2^n$).

3 Discrete Optimisation Approaches

3.1 Overview

Optimisation techniques work either with a single candidate solution or with a population of candidate solutions. Both have been used in the design of Boolean

functions, for example hill-climbing (single)[9] and genetic algorithms (population) [8]. Techniques working with a single solution are usually termed *local* since they consider moving to solutions that are ‘close’ to the current one (i.e. are in the local *neighbourhood*). In contrast, population-based techniques are typically described as *global*. The distinction is not essential (with suitable abstraction population based approaches can also be viewed as local) but we shall follow convention and refer to the simulated annealing-based search we use in this paper as a local search technique.

The optimisation is carried out with respect to some ‘evaluation function’ that measures how ‘good’ a candidate is. The terms ‘fitness function’ and ‘cost function’ are also used. ‘Fitness’ is used most naturally for maximisation problems and ‘cost’ for minimisation problems. To work effectively, an evaluation function must provide sufficient guidance to the search process. In formal terms we require reasonable local smoothness (essentially the fitness/cost values of neighbouring solutions are not too different to the value of the current one). The non-linearity function N_f is reasonably smooth [8] .

We aim to provide balanced functions with high non-linearity (and low autocorrelation). We shall adopt a strategy that starts with a balanced but otherwise random function and moves only to neighbouring solutions that preserve balance. We define the neighbourhood of a balanced function $\hat{f}$ to be all functions $\hat{g}$ obtained from $\hat{f}$ by swapping any two dissimilar values associated with two domain elements $x, y : \mathcal{Z}_2^n$ (we shall refer to the normal representation f and the polar representation $\hat{f}$ as convenient). In formal terms, $\hat{g}$ is in the neighbourhood of $\hat{f}$ if $\exists x, y \in \mathcal{Z}_2^n$ such that

1. $\hat{f}(x) \neq \hat{f}(y)$
2. $\hat{g}(x) = \hat{f}(y), \hat{g}(y) = \hat{f}(x)$ and
3. $\forall z \in \mathcal{Z}_2^n \setminus \{x, y\} : \hat{g}(z) = \hat{f}(z)$

A local search starts at some initial solution $\hat{f}_0$ and advances through a series of neighbouring solutions $\hat{f}_1 \cdots \hat{f}_{end}$. Restrictions may be placed on the relative fitnesses of consecutive $\hat{f}_i$. These determine the nature of the search. Millan et al [9] characterise the fitness of a solution $\hat{f}$ by the pair of values (N_f, AC_f) . For non-linearity a *strong* strategy allows only moves that strictly improve N_f , i.e. we must have $N_{f_{i+1}} > N_{f_i}$. A *weak* strategy requires only that moves do not make the non-linearity worse, i.e we require only $N_{f_{i+1}} \geq N_{f_i}$. Finally it is possible to impose no restrictions on non-linearity. Similar considerations apply to autocorrelation, e.g. a *strong* strategy would require that $AC_{f_{i+1}} < AC_{f_i}$ (since improving here means smaller autocorrelation). Millan et al [9] describe nine search strategies, each combining a strategy for non-linearity and a strategy for autocorrelation. Imposing some element of restriction corresponds to what is generally known as hill-climbing. The most restrictive strategy is (strong, strong). The most permissive is (none, none) — essentially allowing a random walk.

A problem with hill-climbing methods is that they can get stuck in local optima. Some modern heuristic local search techniques work by encouraging

improving moves but allowing (or, with some techniques, forcing) some worsening moves to be accepted as a means of escaping such local optima. Simulated annealing is one such technique and is described in Section 3.4.

In this paper, a minimisation framework is the most natural. We shall optimise with respect to two different cost functions in a two-stage search.

3.2 Cost Functions

Current optimisation work in non-linearity attempts to improve the non-linearity directly. Equivalently (see the definition of N_f in Section 2), it seeks to minimise the cost function

$$\text{cost}(\hat{f}) = WH_{\max}(f)$$

Essentially, the search considers the effect of a move only on those extreme (or near extreme) values of the Walsh Hadamard Transforms $\hat{F}(\omega)$ for the current solution. A more indirect approach can be derived by considering Parseval's theorem below.

$$\sum_{\omega \in \mathcal{Z}_2^n} (\hat{F}(\omega))^2 = 2^{2n}$$

This constrains $WH_{\max}(f) = \max_{\omega \in \mathcal{Z}_2^n} |\hat{F}(\omega)|$ to be at least $2^{\frac{n}{2}}$. It would achieve this bound when for each ω $|\hat{F}(\omega)| = 2^{\frac{n}{2}}$. In practice this bound may be impossible. When some $|\hat{F}(\omega)|$ are greater than this ideal bound, Parseval's theorem ensures that some $|\hat{F}(\omega)|$ must be smaller than it. Thus, it would appear that attempting to restrict the spread of absolute values achieved is well-motivated. This suggests a cost function of the following form:

$$\text{cost}(\hat{f}) = \sum_{\omega \in \mathcal{Z}_2^n} \left| |\hat{F}(\omega)| - 2^{\frac{n}{2}} \right|^R$$

The value R is positive and can be varied. In the experiments reported here we have mostly used $R = 3$. Note that it does not *necessarily* follow that a reduction in our cost function gives rise to an increase in non-linearity but if the range of absolute values is small, then the maximum value will be small too. In section 4.2 we shall further generalise this cost function.

3.3 Calculating Effects of Moves

Every time a move is considered or accepted we must recalculate the values for the various $\hat{F}(\omega)$. It is far more efficient to calculate the changes for each transform using some simplifying equations [9]. If swapping the values of $\hat{f}(x)$ and $\hat{f}(y)$ is a valid move (see Section 3.1) then each Walsh Hadamard Transform $\hat{F}(\omega)$ is changed by an amount

$$\Delta_{\hat{F}}(\omega) = -2\hat{f}(x)\hat{L}_{\omega}(x) - 2\hat{f}(y)\hat{L}_{\omega}(y)$$

$$\Delta_{\hat{F}(\omega)} \in \{-4, 0, +4\}$$

Similar formulae are also available for dealing with changes to correlation elements $\hat{r}_f(s) = \sum_x \hat{f}(x)\hat{f}(x \oplus s)$.

3.4 Simulated Annealing

Simulated annealing is a local search technique that allows escape from local optima. From the current state a move in a local neighbourhood is generated and considered. Improving moves are always accepted. Worsening moves may also be accepted probabilistically in a way that depends on the *temperature* T of the search and the extent to which the move is worse. A number of moves are considered at each temperature. Initially the temperature is high and virtually any move is accepted. Gradually the temperature is cooled and it becomes ever harder to accept worsening moves. Eventually the process ‘freezes’ and only improving moves are accepted at all. If no move has been accepted for some time then the search halts. The technique has the following principal parameters:

- the temperature T
- the cooling rate $\alpha \in (0, 1)$
- the number of moves N considered at each temperature cycle
- the number *MaxFailedCycles* of consecutive failed temperature cycles (where no move is accepted) before the search aborts
- the maximum number IC_{Max} of temperature cycles considered before the search aborts

The initial temperature T_0 is obtained by the technique itself. The other values are typically supplied by the user. In the work described here they remain fixed during a run. More advanced approaches allow these parameters to vary dynamically during the search. The simulated annealing algorithm is as follows:

1. Let T_0 be the start temperature. Increase this temperature until the percentage of moves accepted within an inner loop of N trials exceeds some threshold (e.g. 95%).
2. Set $IC = 0$ (iteration count), *finished* = *false* and *ILSinceLastAccept* = 0 (number of inner loops since a move was accepted) and randomly generate an initial current solution $\hat{f}_{curr}$.
3. while(not *finished*) do 3a-3d
 - a) **Inner Loop:** repeat N Times
 - i. $\hat{f}_{new} = \text{generateMoveFrom}(\hat{f}_{curr})$
 - ii. calculate change in cost
 $\Delta_{cost} = \text{cost}(\hat{f}_{new}) - \text{cost}(\hat{f}_{curr})$
 - iii. If $\Delta_{cost} < 0$ then accept the move, i.e. $\hat{f}_{curr} = \hat{f}_{new}$
 - iv. Otherwise generate a value u from a uniform(0,1) random variable.
 If $\exp^{-\Delta_{cost}/T} > u$ then accept the move, otherwise reject it.
 - b) if no move has been accepted in most recent inner loop then
 $ILSinceLastAccept = ILSinceLastAccept + 1$

- c) $T = T * \alpha$, $IC = IC + 1$
- d) if $(ILSinceLastAccept > MaxFailedCycles)$ or $(IC > IC_{max})$ then
 $finished = true$
- 4. the current value of $\hat{f}$ is taken as the final ‘solution’.

Note that as T decreases to 0 then $\exp^{-\Delta_{cost}/T}$ also tends to 0 if $\Delta_{cost} > 0$ and so the chances of accepting a worsening move become vanishing small as the temperature is lowered.

3.5 Two-Stage Approach

Our technique can now be described very simply:

1. carry out a search using simulated annealing to reach a solution $\hat{f}_{sa}$ with a very low value of the uniform cost function. Calculate the non-linearity $N_{\hat{f}_{sa}}$ of this solution.
2. hill-climb from $\hat{f}_{sa}$ to reach a solution $\hat{f}_{end}$ that is locally optimal with respect to non-linearity (as in [7]). Thus we minimise $WH_{max}(f)$

We view the initial stage as ‘getting in the right area’. Spending too much effort at this stage might actually be counter-productive. Since the initial stage is just a means to an end, we are free to make pragmatic concessions with respect to parameter values. Thus, for example, it is often recommended that N should be equal to the size of the neighbourhood. Even for small problems this would consume much computational effort. We are not bound by this recommendation and choose much smaller N . In fact $n = 400$ was used for all our experiments.

4 Experimental Results

4.1 Comparison of Approaches

In this section we detail the results of applying the technique to the derivation of balanced functions with $n = 8$. The best known bound for the non-linearity is currently 118. The best boolean function demonstrated has non-linearity of 116. Four techniques have been examined: random generation followed by hill-climbing using the traditional cost function; simulated annealing using the traditional cost function; simulated annealing using the new uniform cost function; and simulated annealing using the new cost function followed by hill-climbing using the traditional cost function. We replicated each technique 400-fold. The result of each run is a non-linearity/autocorrelation pair (N_f, AC_f) . For the simulated annealing components the search was terminated after 300 temperature cycles (i.e. inner loops) or else after 50 consecutive cycles had not produced an accepted move. At each temperature cycle 400 moves were considered (as indicated earlier). A temperature factor $\alpha = 0.8$ was used throughout.

Figure 1 shows the results of applying strong hill climbing from randomly chosen initial starting points using the traditional cost function (i.e. maximising

AC_{max}	Non-linearity					
	106	108	110	112	114	116
104	0	1	0	0	0	0
96	0	0	0	0	0	0
88	0	2	1	0	0	0
80	0	5	7	2	0	0
72	1	19	31	6	0	0
64	0	48	78	20	0	0
56	0	25	75	47	0	0
48	0	3	11	18	0	0

Fig. 1. Hill-Climbing with Fitness N_f

the non-linearity N_f directly). No run attained a non-linearity of 114 or more. No run attained an autocorrelation value of 40 or less.

Figure 2 shows the results for applying simulated annealing alone using the traditional cost function. The results are unaffected by following simulated annealing by hill-climbing. Here we see that results are more bunched with respect to non-linearity (all but one have non-linearity of 112). There are a very small number of runs giving values better than for hill-climbing alone.

Figure 3 shows the results of applying simulated annealing alone using the new cost function and Figure 4 shows the results of following this new optimisation by a traditional hill-climb. Now we hit functions with non-linearity of 116 occasionally and some have relatively low autocorrelations.

The results show that the two-stage approach using local optimisation is highly effective. Results have been achieved for non-linearity and autocorrelation that were not obtained using hill-climbing and the traditional cost function. In Millan et al’s more extensive hill-climbing [9] no trial resulted in $N_f = 116$ (when maximising N_f) and no trial resulted in $AC_f = 32$ (when minimising AC_f). As we shall show below, it is possible to improve on the results obtained so far.

AC_{max}	Non-linearity			
	110	112	114	116
80	0	2	0	0
72	0	10	0	0
64	0	59	0	0
56	0	186	0	0
48	0	140	1	0
40	0	2	0	0
32	0	0	0	0

Fig. 2. SA only using Traditional Cost Function (Unaffected by addition of Hill-climbing)

AC_{max}	Non-linearity				
	108	110	112	114	116
56	0	1	0	0	0
48	2	7	35	5	0
40	4	39	158	27	0
32	4	26	79	11	0
24	0	0	2	0	0

Fig. 3. SA Only Using New Cost Function

AC_{max}	Non-linearity				
	108	110	112	114	116
56	0	0	1	7	1
48	0	0	14	56	2
40	0	0	27	176	18
32	0	0	23	64	11
24	0	0	0	0	0

Fig. 4. SA with New Cost Function and Hill-climbing with Traditional Cost Function

4.2 Tuning the Technique

The cost function is a means to an end. Indeed, we chose an initial new cost function for which a decrease did not necessarily correspond to an increase in non-linearity. Its purpose was to get the search to the right area. In using optimisation techniques across a range of problems we have found that experimentation with the cost function frequently produces better results. For this problem we have found that a more effective form of cost function is given by:

$$\text{cost}(\hat{f}) = \sum_{\omega} \left| \left| \hat{F}(\omega) \right| - \left(2^{\frac{n}{2}} + K \right) \right|^R$$

where K can be varied. Figure 5 shows the results of using $K = 4$ and $K = -12$ and $\alpha = 0.9$ (other parameters as before). The effects of such tuning are marked. The technique has produced values that were not achieved by any hill-climbing strategy (or indeed by the work reported here so far). The results are all the more remarkable since autocorrelation was effectively ignored as (*a conscious*) part of the search. The values of K clearly influence which parts of the design space are reached. For $K = -12$ the search has tended to find solutions with lower non-linearity than for $K = 4$ but generally lower (and so better) autocorrelation. Also, examples of functions with autocorrelation of 16 have been found easily by the search. Small-scale experimentation has been carried out with various K values and the results are presented in Figure 6. The results presented here are for illustration only. More extensive experimentation is required. We leave this as future work.

	Non-linearity					
AC_{max}	112	114	116	112	114	116
56	0	0	0	0	0	0
48	2	5	13	0	1	3
40	9	68	80	11	16	8
32	29	74	115	41	42	24
24	1	1	3	201	6	5
16	0	0	0	42	0	0
	K=4			K=-12		

Fig. 5. SA with New Cost Function followed by Hill-climbing with Traditional Cost Function for $K = 4, -12$ (400 runs)

	Non-linearity											
AC_{max}	112	114	116	112	114	116	112	114	116	112	114	116
48	0	0	0	0	1	0	0	0	1	0	0	0
40	3	2	2	3	2	3	2	5	2	2	2	1
32	19	10	6	15	8	2	15	9	3	11	12	6
24	45	2	0	53	2	0	51	2	0	44	3	1
16	11	0	0	11	0	0	10	0	0	18	0	0
	K=-14			K=-12			K=-10			K=-8		

	Non-linearity											
AC_{max}	112	114	116	112	114	116	112	114	116	112	114	116
48	0	0	1	0	3	2	0	8	5	5	12	1
40	0	5	13	3	19	20	6	32	15	12	43	1
32	2	27	42	5	17	27	3	16	15	6	19	1
24	0	0	10	1	1	2	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0
	K=-6			K=-4			K=-2			K=0		

Fig. 6. Results for $n = 8$ with various K values (100 runs)

4.3 Results for Higher Values of n

We have applied the technique with the tunable cost function to the synthesis of Boolean functions with $n = 9, 10, 11, 12$ variables. The number of inner cycles N was 200 (except for $n = 12$ where 400 cycles were carried out) with 400 moves considered in each cycle as before. The overall aim of the work was to generate highly non-linear Boolean functions (low autocorrelation values obtained were accidental). Figure 7 summarises how our technique compares with the best results obtained so far by other techniques (both constructive techniques and heuristic optimisation approaches) with respect to non-linearity.

For lower values of n our technique has no difficulty in equalling the best produced so far by any technique. We have not yet carried out extensive experiments with respect to parameter variation and make no claims to optimality of our results above. Experience with other problems has shown that cost function variation generally leads to improvements. Hard and fast comparisons with other

Method	4	5	6	7	8	9	10	11	12
Lowest Upper Bound	4	12	26	56	118	244	494	1000	2014
Best Known Example [11,3]	4	12	26	56	116	240	492	992	2010
Bent Concatenation	4	12	24	56	112	240	480	992	1984
Genetic Algorithms [8]	4	12	26	56	116	236	484	980	1976
Our Simulated Annealing	4	12	26	56	116	238	484	984	1990

Fig. 7. Comparing the Non-linearity of Balanced Functions

optimisation work is difficult since the simulated annealing is likely to be more computationally complex (here 80000 function evaluations before hill climbing). Nevertheless, our techniques seems to compare favourably with genetic algorithms for the higher values of n (work performed at the request of the referees). The results are encouraging and indicate that local search has good potential as a synthesis technique.

For reference purposes we include below the results of our experiments for $n = 9, 10, 11, 12$. For $n = 9, 10$ we provide the results of (small-scale) random function generation followed by hill-climbing with respect to non-linearity (this is far better than random generation alone as demonstrated in [7]). For $n = 11, 12$ we simply present the results obtained by our technique to allow comparisons by future researchers.

AC_{max}	Non-linearity	
	236	238
72	3	1
64	16	13
56	97	60
48	125	76
40	5	4

Fig. 8. 400 runs for $n = 9, \alpha = 0.9$ and $K = -12$

5 Conclusions and Further Work

The work reported in this paper has shown how *local* search can be used to generate boolean functions that have both high non-linearity *and* low autocorrelation. Further work is needed to determine optimal parameter values for the technique. The values reported here are for illustration only. Indeed since the efficacy of our technique is significantly affected by parameter values we propose to investigate the use of *adaptive cost functions*. With this approach the search is continually monitored and parameters varied dynamically. The parameters for the annealing algorithm have been chosen to allow a reasonable amount of experimentation. The value $N = 400$ is extremely small compared with recommended values (and the larger the number of Boolean variables n the worse is

AC_{max}	Non-linearity				
	224	226	228	230	232
136	1	1	1	0	0
128	0	1	5	1	0
120	4	3	8	1	0
112	4	6	25	3	0
104	6	14	51	11	0
96	3	9	87	23	4
88	1	5	55	26	2
80	0	0	18	15	1
72	0	0	0	3	2

Fig. 9. 400 randomly generated functions followed by Hill-climbing for $n = 9$

AC_{max}	Non-linearity		
	480	482	484
104	0	1	1
96	0	0	8
88	0	1	23
80	0	1	41
72	0	0	24

Fig. 10. 100 runs for $n = 10, \alpha = 0.9$ and $K = -16$

AC_{max}	Non-linearity				
	464	466	468	470	472
192	0	0	0	1	0
184	0	0	0	1	0
176	0	0	1	1	0
168	1	0	4	2	2
160	1	2	5	2	4
152	0	0	3	6	3
144	0	0	11	11	9
136	0	2	3	5	9
128	0	0	0	3	5
120	0	0	0	2	1

Fig. 11. 100 randomly generated functions followed by Hill-climbing for $n = 10$

the discrepancy). Also, the values used for the cooling factor α (0.8 and 0.9) are very small compared with the bulk of successful annealing work. The results so far show considerable promise but further work is needed to tune the technique effectively. The use of global optimisation techniques such as genetic algorithms in conjunction with the new cost function would also seem worthy of investigation. We aim to extend the local simulated annealing approach to incorporate other desirable cryptographic properties.

AC_{max}	Non-linearity		
	980	982	984
160	0	1	0
152	0	0	0
144	0	2	2
136	0	7	2
128	4	11	7
120	1	23	6
112	2	19	5
104	0	5	3

Fig. 12. 100 runs for $n = 11, \alpha = 0.9$, $K = -24$ and $R = 2.5$

AC_{max}	Non-linearity	
	1988	1990
212	1	0
204	0	1
196	0	0
188	3	1
180	1	0
172	1	1
164	1	0

Fig. 13. 10 runs for $n = 12, \alpha = 0.9$, $K = -28$ and $R = 3.0$

As is typically the case with heuristic techniques, augmenting the basic approach with hill-climbing has been found to be an excellent idea. The simulated annealing simply provides a means of locating good areas from which to hill-climb. We have found that adopting a two-stage strategy is of use in other security problems too.

Perhaps the most important point of this paper is that the cost function matters. The authors have recently applied optimisation techniques to a variety of security related problems. Experimentation with cost functions has typically led to better results (in some cases a radical improvement is obtained). We would recommend such experimentation to all.

We are currently working on the use of optimisation techniques to derive cryptographic artifacts which satisfy excellent publicly stated criteria but which also satisfy secret malicious criteria!

Acknowledgements. We would like to thank the members of the SEMINAL network (EPSRC grant GR/M78083) for their encouragement in carrying out this work.

References

1. W.S. Forsyth and R. Safavi-Naini. Automated Cryptanalysis of Substitution Ciphers. *Cryptologia*. vol. XVII. No 4. 1993. pp407–418.
2. J.P. Giddy and R. Safavi-Naini R. Automated Cryptanalysis of Transposition Ciphers. *The Computer Journal*, (XVII)4, 1994.
3. X.-D Hou. On the Norm and Covering Radius of First-order Reed-Muller Codes. *IEEE Transactions on Information Theory*, 43(3). May 1983. pp.354–356.
4. S. Kirkpatrick and C. D. Gelatt, Jr. and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, May 1993. pp. 671–680
5. Lars R. Knudsen and Willi Meier. Cryptanalysis of an Identification Scheme Based on the Permuted Perceptron Problem. In *Advances in Cryptology Eurocrypt '99*, Springer Verlag LNCS 1592 pp. 363–374
6. Robert A J Mathews. The Use of Genetic Algorithms in Cryptanalysis. *Cryptologia* (XVII)2, April 1993. pp187–201
7. William Millan, Andrew Clark and Ed Dawson. An Effective Genetic Algorithm for Finding Highly Non-linear Boolean Functions. In *Proceedings of the First International Conference on Information and Communications Security*, 1997. Springer Verlag LNCS 1334. pp149–158
8. William Millan, Andrew Clark and Ed Dawson. Heuristic Design of Cryptographically Strong Balanced Boolean Functions. In *Advances in Cryptology EUROCRYPT'98*, Springer Verlag LNCS 1403. pp.489–499
9. William Millan, Andrew Clark and Ed Dawson. Boolean Function Design Using Hill-climbing Methods. *Australian Conference on Information Security and Privacy* (ACISP). 1999. pp.1–11
10. National Bureau of Standards. NBS FIPS PUB 46. Data Encryption Standard. 1976.
11. N.J. Patterson and D.H. Weidemann. The Covering Radius of the $(2^{15}, 16)$ Reed-Muller Code is at Least 16276. *IEEE Transactions on Information Theory*, 29(3). May 1997. pp.1025–1027.
12. Richard Spillman and Mark Janssen and Bob Nelson and Martin Kepner. The Use of A Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers. *Cryptologia* (XVII)1. April 1993. pp. 187–201

A Novel Engine for Various Intrusion Detection Methods

Zhu Hui and T.H. Daniel Tan

Information Systems Research Lab
School of Electrical & Electronic Engineering
Nanyang Technological University
Singapore, 639798
{p144778961, ethtan}@ntu.edu.sg

Abstract. In this paper, we describe a novel Sensor-Based Intrusion Detection Engine – SenIDS, which can process different security-related data types with various intrusion detection methods. With SenIDS, we can integrate the misuse intrusion detection method and anomaly intrusion detection method into a single structure and algorithm. SenIDS constructs a framework for intrusion detection, which includes sensors (a complex structure including a sub-program and fields representing desired event record source, the user and program activity, variable names and values, etc.) The sensor structure possesses greater ability to detect and handle a variety of complex intrusion scenarios. The intrusion detection rule consists of one or more sensors. Alarms in such rules can be triggered by various intrusion instances. Such processes on these sensors and rules are managed by a Trigger Engine automatically. The Trigger Engine can manage different kinds of sensors and rules for triggering. This enables SenIDS to integrate different intrusion detection methods.

1 Introduction

Presently, network intrusions and computer-initiated abuse have increased dramatically due to the popularity of the Internet and the transparency of the network users and technology. In response, many tools have been developed to secure networks, among which are intrusion detection systems (or IDS). Intrusion Detection systems are software systems to detect the intrusion and anomalous activities through the analysis of the security-related data.

Currently, IDSs usually process network management or security-related data such as audit trail data, network traffic, application log, system data collected by dedicated programs, etc. to detect intrusions. According to the methods, intrusion detection technologies can be broadly categorized into anomaly detection and misuse detection.

Anomaly IDSs attempt to establish normal behavior profiles for users, hosts, programs and processes of a network. Significant deviations of activities to the established behavior profiles are broadly classified as instances of intrusion. Several IDSs, like HAYSTACK [1], MIDAS [2], employ such an anomaly detection approach. Recently, Lane *et al.* [3] modeled the user behavior with the sequence (an ordered, fixed-length set of temporally adjacent actions). Also, Ko *et al.* [4] developed a specification-based language to specify the intended behavior of security-critical programs, with specifications indicating the desired activity pattern of the

program. The premise here is that every activity beyond the specification is likely to be an attack. The main difficulties in anomaly detection are the feature sets (which are used to generate the profile) must have the essential capability to differentiate malicious activities from the normal behavior.

Misuse IDSs compare the sub-set of specific security-related data with predefined intrusion signatures to indicate such intrusions. For example, Lindqvist *et al.* [5] used a *Production-Based Expert System Toolset* for misuse detection. Mounji [6] developed a rule-based language (*RUSSEL*) to describe the signatures and rules of intrusions. Ilgun [7] designed USTAT to analyze the system state transition to detect the intrusions. Colored Petri Nets is used by Kumar [8] to provide a pattern matching-based computational model. The main limitations are that (a) unknown intrusions are hard, if not impossible to detect, and (b) intrusion signatures are difficult to construct.

This project aims to integrate both anomaly and misuse detection methods into a single framework. The SenIDS intrusion detection engine has the ability to process a variety of security-related data with a combination of various algorithms. With SenIDS, we can employ anomaly and misuse detection methods to process different data types to detect a single intrusion scenario. Here, we introduce the concept of sensor. This sensor is a complex structure with a sub-program and fields to represent user or program activities, variable names and values, etc. The detecting rule consists of one or more sensors. Such a rule has the enhanced ability to detect reliably various intrusion scenarios. The Trigger Engine (or TE) is employed to activate the sensors with security-related events. The TE can create, save, and delete the triggered sensors and rules.

2 Sensor-Based Intrusion Detection Engine – SenIDS

SenIDS is an intrusion detection engine. It has enhanced ability to process various security-related data types with various detecting algorithms.

The security-related data, including audit trail data, network packet, application log, and system data, etc. will be collected by data collector and fed to the SenIDS. In order to process different kinds of security-related data, these data must be convert to a standard format before fed to SenIDS. The standard format (like the format proposed in [9]) consists of variable-length fields; each field consists of a field name and a value. The field name must be predefined for processing by the SenIDS. Every event record type should have one field with the name „source“ to indicate the type of the data, such as audit trail and network packet. SenIDS can process different types of security-related data according to this field.

These event records are transferred to the selected sensors for processing. The sensors are made up of some fields and a sub-program; one or more sensors compose a detecting rule, which is used to detect an intrusion instance. A Trigger Engine will be employed to manage the event records, sensors and rules. The Trigger Engine can pre-select the waiting sensors (the sensor that is waiting to be triggered) to trigger, try to trigger the selected sensor with event record, update the waiting sensor, and delete the waiting sensors. Also, the Trigger Engine can create the triggered rules, update the triggered rules and delete the triggered rule. The Trigger Engine manipulates the sensors and rules according to their type and their „return“ instruction automatically. When certain rule is fully fired, the TE will raise an alarm.

3 Sensor and Rule

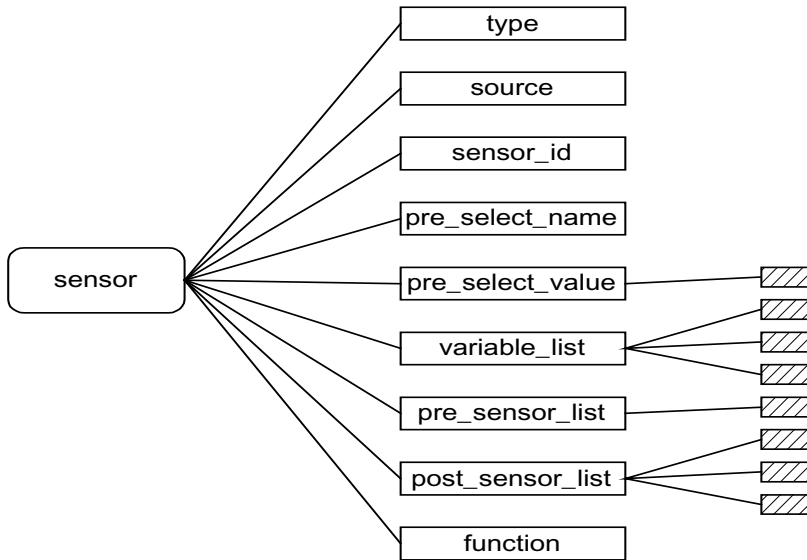


Fig. 1. Structure of Sensor

The approach taken in SenIDS is based on the sensor concept. Sensor is a complex structure including a sub-program and fields representing desired event record source, the user and program activity, variable name and values, etc. Every detecting rule consists of one or several sensors to detect an intrusion instance.

3.1 Structure of Sensor

The sensor of SenIDS is configured as a data structure, containing several fields (see Fig. 1). The fields include:

- *Type*: indicates the type of the sensor. (We will talk about the sensor type in next section)
- *Source*: indicates the type of the event record, which is desired by the sensor. TE will compare this field with the *source* field of the event record to pre-select sensors for triggering.
- *Sensor_id*: represents the unique id of the sensor.

- *Pre_select_name* and *pre_select_value*: are the pre-selection criteria. *Pre_select_name* is the field name that is corresponding to one field name of the event record. *Pre_select_value* is the desired value of that field that should be the same as the value of that field in the event record. These two fields and *source* field are used for pre-selection. Pre-selection aims to shorten the triggering process time by pre-select the appropriate sensors for triggering instead of trying to trigger every sensor with the event record. When the system event occurs, the TE will try to find the appropriate sensors to trigger, whose *source* field is the same as the event record and the *pre_select_name* and *pre_select_value* have the corresponding field in event record. *Pre_select_value* is a list that includes all possible values to be matched.
- *Variable_list*: describes an array of variable names and values. The *variable_list* stores the information generated by the triggering process of sensor. This information will be used to trigger the other sensors. Different sensors can transfer information to each other by *variable_list* field of sensors.
- *Pre_sensor_list* and *post_sensor_list* describe the *sensor_ids* of sensors that must be triggered before or after the current sensor is triggered. With these two fields, sensors can be linked together to construct rules for detecting complex intrusion instances.
- *Function*: is the name of a sub-program that will process the event record data to trigger the sensor. The sub-program is stored in a separated file. With this sub-program, detecting „rules“ in the sensor structure can be excluded. It, therefore, presents a more flexible means to write „rules“ for triggering the sensor in a separated file without the need for a special language. We can construct a function library that contains the common functions. These functions can be employed to construct the sub-program of sensors to eliminate the duplicated programming.

With these fields, the sensor possesses greater ability to detect and handle a variety of complex intrusion scenarios.

3.2 Intrusion Detection Rule

The intrusion detection rule consists one or several sensors that are designed to detect a single misuse, violation or intrusion activity. The intrusion detection rules can represent the intrusion signatures; can be used to establish behavior profiles and detect the anomaly; can be designed to detect violations and other malicious activities. The intrusion detection rules are stored in the intrusion detection *Rule Table*, which will be consulted for the rules and sensors by Trigger Engine.

3.3 Classification of Detecting Rule

The data structure design approach enhances the sensor's ability to be used for detecting different types of intrusions (both simple and complex). With sensors, we can construct the following types of rules to be employed in SenIDS.

3.3.1 Intrusion Signature

Intrusion signature specifies the features, conditions, arrangements and interrelationships among events that lead to a break in or other misuse [8]. The intrusion signature is abstracted from the past attacks, known weakness (vulnerabilities), and security policy. Intrusion signature is employed by misuse intrusion detection system to compare with the security-related data in order to detect intrusion activities. In SenIDS, the intrusion signature can be constructed using a sequence of sensors. Usually, the intrusion consists of a sequence of steps, which leave a series of system events in the log. To detect such an intrusion, a sequence of sensors to scan the system events is employed, which compose intrusion detection rule. The *pre_sensor_list* and *post_sensor_list* fields of sensor represent the relationship between sensors in an intrusion signature. Sensors in the same intrusion signature can transfer messages between each other via the *variable_list*. In this way, the combinations of these sensors (in various orders and sequences) can be utilized to represent complex intrusion signatures.

3.3.2 Behavior Profile

Anomaly intrusion detection method usually establishes the behavior profiles of users, programs and system activities, etc.; try to detect the significant deviation of the real-time activity to the established profile to identify the intrusion. Such behavior profiles of users, systems, programs and processes can also be defined from sensors. The sensor can collect profile values, which can be stored in *variable_list* and updated to the triggered rule. We can define the threshold values in the sub-programs of such sensors; when exceed, that particular sensor will raise an alarm.

3.3.3 System Policy and Anomaly Alarm

Some activities, which are normal in other sites, may violate your company's policy. Sensors can be used to represent the system policy unique for different sites. They can be used to scan for after-the-fact violations of system policy. For example, a sensor can be easily designed to search for login events that occurred later than 10:00PM. Anomalies can also be detected with the sensors. For example, a sensor can be constructed to trigger by continuous failed logins within a three-minute window.

3.3.4 Others

The sensor provides us a framework to process security-related data. With sensors, we are free to employ any algorithms to process the event records. For example, we can design a sensor to find the new user of the system and we can also use sensor to collect the information of how many users have logged into the system for a day.

4 Algorithms of SenIDS

As mentioned earlier, in SenIDS, the Trigger Engine (or TE) controls the sensors and rules. This TE is the major component of SenIDS. The TE generates and manages the

wait_sensor table (WS table) and *triggered_rule table (TR table)*. *Wait_sensor table (WS table)* stores the sensors that are waiting to be triggered. These sensors include the first sensors of every rules that are expected to be triggered to detect a new suspect intrusion instance and the sensors of the triggered rules that are expected to be triggered in order to fully detect the current intrusion instance. *Triggered_rule table (TR table)* stores the triggered rules; each triggered rule corresponds to a suspect intrusion instance. Every record in the *TR table* is a suspect intrusion instance, which includes the identification number of the instance, variable values received from the previous triggered sensors, the event records that triggered previous sensors, etc. TE can pre-select the suitable waiting sensors in *WS table* to be triggered; try to trigger the waiting sensors with the event record; save the triggered rules' information into the *TR table*; or delete the triggered rules and sensors.

4.1 How Trigger Engine Works

When SenIDS executes, the TE will read the first sensor of each rule in *Rule table* and write them to the *WS table*. These sensors are waiting to be triggered in order to find a new suspect intrusion instance. When one of these sensors is triggered, we think a suspect intrusion instance is raised; TE will generate a new triggered-rule in *TR table*, and add the next sensor of the same rule to the *WS table* that will wait to be triggered in order to fully detect the instance.

When a system event arrived (Fig. 2.), TE will try to trigger each waiting sensor in *WS table*. This is done by, firstly, TE pre-selecting the suitable sensors according to the *source*, *pre_select_name* and *pre_select_value* fields of the sensors. Then, the TE will run the sensor sub-program with the system event record to determine if the sensor can be triggered.

If the sensor cannot be triggered, TE will simply ignore the current sensor and try to select another sensor to trigger. If the sensor is triggered, TE will process the sensor and rule according to the type of the sensor and the return message of the sub-program. The return message of the sensor sub-program includes an *action* field to instruct the TE how to process the current sensor. For example, if it is a multi-sensor intrusion signature, it will be sent to the *TR table* along with the system event and the return messages of the sub-program. At the same time, the sensors in the *post_sensor_list* of the triggered sensor will be written to the *WS table*, and wait to be triggered in order to fully detect that instance. We will talk about the TE actions according to the different return messages in next section.

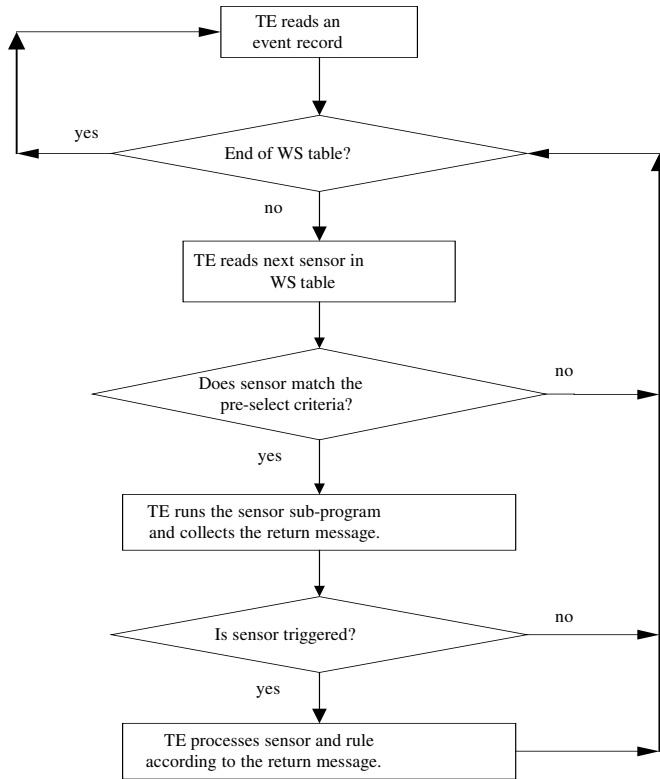


Fig. 2. Flow Chart of TE

4.2 Messages between Sensors and TE

The sensor can send messages to TE, as well as to other sensors of the same rule. When TE runs the sub-program, the sub-program will return some message back to TE. The message includes *action* to instruct TE what to do about the current sensor and some variable values that should be save to the *variable_list* for the use of other sensors of the same rule.

The *action* returned by the sensor sub-program is an instruction that instructs TE what to do about the current sensor. Such *actions* are „create“, „cleanup“, „fail“, „active“, „alarm“, „next“, „reset“, etc. „Create“ instructs the TE to create a new triggered rule in *TR table*. „Cleanup“ instructs the TE to delete the current triggered rule and sensors. „Fail“ tells the TE that the current sensor is not triggered by that event. „Alarm“ records an detected intrusion event. „Active“ instructs the TE to maintain the current waiting sensor in the *WS table* and update the information in the *TR table*. Other *action* values can be used in different situations.

The variable values returned by sub-program are used to transfer information between sensors of the same rule. They may include, for example, the time-stamp when the specific event occurs; the object file or system resource that should remain unchanged throughout the same rule, and other information to unify the triggering process in the same rule.

4.3 Life-Span of Triggered Rule

In order to manage the population of the triggered rules and manipulate different types of rules, we must consider the life-span of the triggered rules. The life-span of the triggered rules refers to the generation and termination of triggered intrusion rules in the *TR table*. It is very important for manipulating different rule types.

When the first sensor of a rule stored in *WS table* is triggered, the TE will create a new record in the *TR table*; this indicates a new suspect intrusion instance arising from the triggered rule.

The TE will terminate the triggered rule when the following condition(s) apply:

- The triggered rules in *TR table* will be fully triggered, when all the sensors of that rule are triggered. This rule will be deleted after TE reported the intrusion. The related waiting sensors in *WS table* will also be deleted.
- The triggered rules in *TR table* will terminate if a sensor sub-program returns the „cleanup“ message to TE. This happens when a sensor determines that the current triggered rule would not be completed forever. The triggered rule in *TR table* and the related sensors in *WS table* will be deleted.
- For statistical collection rules, the sensor of the rule itself determines to terminate. If the threshold is exceeded, the sensor will send messages with *action* „alarm“ to raise an alarm and instruct the TE to terminate the current rule and sensors. Otherwise, the sensor will send *action* „active“ to TE, which instructs TE to update the same waiting sensor in *WS table* and update the corresponding triggered rule in *TR table*.
- The policy rules will determine the termination by itself. Sensors of policy rule will send „alarm“ to the TE to indicate a violation and instruct TE to terminate the current rule and sensors. Otherwise it can also determine to „cleanup“ without a violation.
- For the rules to establish behavior profile they will keep alive infinitely; however, the variables will change every time the sensor is triggered. And the sensor will send „active“ to instruct TE to update the current sensor in *WS table* and rule in *TR table*. However, they can raise an alarm, when an anomaly is detected.
- For all of the rules, a time span is defined; if the triggered rules in the *TR table* stay longer than the predefined time span, the TE will terminate that rule and related sensors automatically.

In this way, the sensor population will not grow uncontrollably. In this regard, sensors should be created with its termination as part of the design process.

5 SenIDS Rule Examples

SenIDS has been developed using Perl on SUN Solaris 2.5. Currently, the security related data are collected in audit trail produced by the Solaris Basic Security Module (BSM). The system can also be developed using C or C++ for performance consideration.

An example of a rule for detecting failed authentication attempts within a certain time span is discussed here. An alarm will be raised when a user fails in the login authentication process when attempts are made more than *Y* times within *Z* seconds.

Such an authentication failure is defined as entering an invalid username and password to a *login*, *rlogin*, or *su* process. This rule is implemented in a sensor descriptor (see Table 1).

In Table 1, *Type* „statistic“ indicates the sensor is used to collect statistics. *Source* „audit_trail“, *pre_select_name* „header_event_id“ and *pre_select_value* „login, rlogin, su“ is used to pre-select the sensors to be triggered by the TE. From these fields, we can see that just the audit_trail event record, whose value of field „header_event_id“ is one of „login“, „rlogin“ or „su“, can trigger the sensor. The *variable_list* items „time1“ and „number“ store the values generated each time the sensor is triggered. *Function* „fail_auth“ is the sub-program name that is used to trigger the sensor with the event record. Because this sensor is a single-sensor rule, so the *pre_sensor_list* and *post_sensor_list* contain no *sensor_ids*.

The following program code illustrates the sub-program named „fail_auth“ of the sensor. The hash data structure *%standard{}* stores the security-related event record fed by data collector in standard format. The hash data structure *%return_message{}* is returned by the sub-program to TE, among which *\$return_message{action}* will send to the TE to instruct TE what to do about the current sensor and related triggered rules. *\$return_message{time1}* returns the time when the first failed login happens. *\$return_message{number}* returns the number of failed login within the time span.

Table 1. Sensor for Detecting Failed Authentication

Name	Value
Type	statistic
Source	audit_trail
Sensor_id	15
Pre_select_name	header_event_id
Pre_select_value	login, rlogin, su
Variable_list	time1, number
Pre_sensor_list	0
Post_sensor_list	0
Function	fail_auth

```

sub fail_auth
{if (($standard{header_event_id} = „AUE_login“ or
    $standard{header_event_id} = „AUE_rlogin“ or
    $standard{header_event_id} = „AUE_su“ )and
    ($standard{return_error_value}=„INVALID_USER“ or
    $standard{return_error_value}=„INVALID_PWD“))
    {if ($number = „ “){
        $number=1;
        $time=$standard{header_data_time};
        $return_message{action}=„create“;
        $return_message{time1}=$time;
        $return_message{number}=$number;
        return \"%return_message;

```

```

elseif($number+1>=Y) {
    $return_message{action}="alarm";
    $return_message{alarm}="Alarm,           Failed
authentication attempts are more than Y
times";
    return \%return_message;}
elseif($standard{header_data_time}-$time1>Z) {
    $return_message{action}="cleanup";
    return \%return_message;}
else{$return_message{number}=$number+1;
    $return_message{action}="active";
    return \%return_message;}
}
else{$return_message{action}="fail";
    return \%return_message;}
}

```

In this example, when an event record comes to the TE, TE will pre-select the sensors in the *WS table* to trigger by comparing the sensor *source*, *pre_select_name* and *pre_select_value* with the corresponding field of event record. The sub-program of the selected sensor will run by the TE with the event record.

We assume that this failed-login detecting sensor is selected. The sub-program „fail_auth“ will run by the TE. It firstly checks if the event record is the failed „login, rlogin, su“. If not, it simply returns the *\$return_message{action}="fail"* to TE, which indicates that the current event record cannot trigger the sensor; in this case, TE just find another candidate sensor in *WS table* to trigger. Otherwise, „fail_auth“ will try to determine if the sensor is triggered for the first time, or the sensor exceeds the predefined the maximum number *Y*, or the sensor exceeds the predefined time span *Z*. For the first-time triggered sensor, „fail_auth“ will send *\$return_message{action}="create"* to TE; according to the type of the sensor, TE will create a new triggered rule in *TR table* and update the sensor in *WS table*. If the sensor is triggered more then *Y* times, the „fail_auth“ will send *\$return_message{action}="alarm"* along with the alarm message to TE, which instruct TE to raise an alarm and delete the current triggered rule and waiting sensor in *TR table* and *Ws table*. If the sensor exceeds the time span *Z*, it will return *\$return_message{action}="cleanup"* to tell the TE to clean the current triggered rule in *TR table* and all the related waiting sensors in *WS table*. If the sensor has been triggered before but not yet exceeds the number *Y*, TE just updates the corresponding triggered rule in *TR table* and the waiting sensor in *WS table*.

This is a simple example rule that contains just one sensor. From this example, we can see that TE just manages the rules and sensors; it does not care about how to process the event record. Much of the work must be done by the sensor sub-program. It gives us greater abilities to construct more powerful sensors to detect complex intrusion scenarios. The structure of the sensor and the function of TE make it possible to integrate the misuse intrusion detection and anomaly intrusion detection into a single intrusion detection engine.

6 Conclusions and Future Work

In this paper, we propose the SenIDS system that employs a sensor framework to implement and achieve different types of intrusion detection. With these sensors, SenIDS can effectively combine misuse intrusion detection and anomaly intrusion detection. SenIDS incorporates different kinds of sensors to detect various intrusion scenarios.

The challenge of the SenIDS system will be to design comprehensive sensor libraries. Such libraries should incorporate intrusion signatures, security policies, user profiles, processes profiles, etc. Well-designed rules that represent user and process profiles of the activities will enhance the system abilities to detect unknown intrusion scenarios. Various security-related data types, such as application log and network packet, will be integrated into our system in the near future.

References:

1. S. E. Smaha. HAYSTACK: An Intrusion Detection System. In IEEE Fourth Aerospace Computer Security Applications Conference, Orlando, FL, December 1988.
2. M.M. Sebring, E. Shellhouse, M.E. Hanna and R.A. Whitehurst. Expert System in Intrusion Detection: A Case Study. The 11th National Computer Security Conference, Baltimore, MD, October 1988.
3. T. Lane, C. E. Brodley. Sequence Matching and Learning in Anomaly Detection for Computer Security. In Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, pp 43-49. 1997.
4. C. Ko, M. Ruschitzka, and K. Levitt. Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach. In Proceedings of the 1997 IEEE Symposium on Security and Privacy, pp. 134-144, 1997.
5. Ulf Lindqvist and Philip A. Porras. Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST). In Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, California, 1999.
6. Abdelaziz Mounji. Language and Tools for Rule-Based Distributed Intrusion Detection. PhD thesis, d'Informatique, Facultes Univsitaires Notre-Dame de la Paix, Namur, Belgium, September 1997.
7. K. Ilgun. USTAT: A Real-Time Intrusion Detection System for UNIX. In Proceedings of the 1993 IEEE Symposium on Security and Privacy, pages 16-29, 1993.
8. Sandeep Kumar. Classification and Detection of Computer Intrusion. PhD thesis, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, August 1995
9. M. Bishop. A Standard Audit Log Format. In Proceedings of the 1995 National Information Systems Security Conference. Baltimore, Maryland, pp. 136-145, October 1995.

Construction and Categories of Codes

G.R. Blakley, I. Borosh, and A. Klappenecker

Department of Mathematics, Texas A&M University, College Station,
TX 77843-3368, USA.

{`blakley`, `borosh`, `andreask`}@math.tamu.edu

Abstract. Blakley and Borosh introduced a general theory of codes, encompassing cryptographic and error control codes among others. They explored the properties of such general codes with methods from relational algebra and set theory. We provide a categorical point of view, which leads to new constructions of codes. We also exhibit a Jordan-Hölder type theorem and a Schreier refinement technique.

1 Introduction

In the late twentieth century a vast proliferation of codes occurred. Many new cardinalities became common, especially large finite or infinite. Many new arithmetics – infinite as well as finite – could be found in the newly introduced arithmetic-based codes. Hilbert spaces are as integral to the theory of quantum error control as Hamming spaces to classical error control. But many new codes arose without arithmetic, amounting to mere codebooks or databases.

Codes with no encode process, codes with no decode process, codes which encode every plaintext symbol into billions of different codetext expressions are now famous and widely used, as are codes which decode every codetext expression into every plaintext symbol.

Commerce has made ISSN, ISBN, UPC commonplace. Locks are codes, phonebooks are, genomes are, cash register receipts are, codes replace telephone wires, tollbooths, signatures.

This is not metaphorical talk. Every one of these objects is a code in a strict mathematical sense. And one realization that emerges from this mathematical view of codes is the profound importance of structural considerations. Codes have shapes, just as molecules have shapes. And the designer of codes has a larger repertory of kinds of structures to draw upon than an organic chemist. Moreover, these kinds of structures can be usefully described and combined by the methods of universal algebra, as adumbrated in [3]. But they also lend themselves to treatment by category theory, as will become clearer below.

After defining and visualizing precodes and codes, we introduce the corresponding categories in Section 4. We obtain products, limits, and colimits in the usual way in Sections 5 and 6, and then give some examples. The subprecodes of a precode form a lattice in a natural way. Section 8 discusses some of its properties and, in particular, establishes a Jordan-Hölder-Schreier theory for it. This result suggests a unified view of several cryptanalytic methodologies. We conclude with comments on this need for further results.

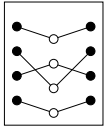
2 Visual Presentation of Precodes and Codes

The code definition given in the general theory [2] of codes goes as follows. A **precode** is a list (P, C, e, d) whose entries are a set P of plaintext symbols, a set C of codetext symbols, an encode relation $e \subseteq P \times C$, and a decode relation $d \subseteq C \times P$. A **code** is a precode for which the composite relation $de \subseteq P \times P$ is subdiagonal, i. e. contains only pairs of the form (p, p) .

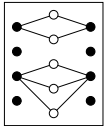
We follow [2] and introduce a graphical representation of precodes and codes called a **strip chart**. The items P, e, C, d, P are represented by five columns of marks. The marks in the three ‘symbol’ columns P, C, P are at various heights – different heights signifying different elements. The marks in the two ‘relation’ columns e, d are undirected line segments treated as if they were arrows going from left to right.

For example the first strip chart below represents the popular notion of a code, as a pair e, d of bijections between P and C . The bijection d going from the codetext symbols C (the set of open rings in the third column) to the plaintext symbols P (the set of blobs in the fifth column) is the inverse of the bijection e going from the blobs of P in the first column to the rings of C in the third column. Clearly each action of e moves a first-column blob b to a third-column ring r , and then d takes this ring to a fifth-column blob at the same height as the original blob. In other words, de takes each blob to itself.

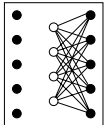
It is worth becoming acquainted with the strip charts below. They give a weak foreshadowing of the huge variety of codes already in use. And they set the stage for the purposeful use of abstract structure to produce novel codes of yet widely different structural types which the general theory of codes can supply for various information-related investigations or activities.



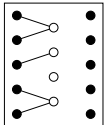
In the figure at left, the encode e is a bijection from P to C , the decode d is its inverse function, a bijection from C to P . Gray codes, key settings of Caesar ciphers, RSA, DES, AES, some commercial codebooks and Gödel numbering are among the many example of this matched-pair-of-bijections type of code.



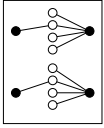
The code at left has a decode d which is a function (i. e., many-to-one relation). Its encode e is the converse of d (whence it is a one-to-many relation). There are many such codes, including some codebooks with homophones, the calculus (in which encode is antidifferentiation of a function and decode is differentiation, secret sharing schemes, and hash function codes (hashes are decodes, and their converses are encodes)).



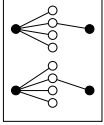
The code at left has an empty encode relation $e = \emptyset$, and a full decode (an all-to-all relation) d . Hence de is empty, so this strip chart does, indeed, present a code. The Diffie-Hellman key exchange [6] is often a pair of codes of this one-way ‘encodeless’ type. The genetic code is similar, but less extreme. Its encode is empty, and its decode is a function.



The code at left has an empty decode relation $d = \emptyset$, and an encode relation e which is a function. Clearly $de = \emptyset$. The Purdy high security login [9] is a code of this one-way ‘decodeless’ type. Other such examples are hash functions when viewed as one-way objects.



decode.



The code at left has an injective decode d , and a one-to-many encode relation e . It amounts to a variant of the magnetic-strip-card-key code, which large hotels use to give guests entry to their rooms. Each room is ‘encoded’ as a large set of bit strings, only one of which is valid today. A card with this string ‘decodes’ the room door to open today – thus revealing the encoded room. Cards with any other bit string don’t open it. And tomorrow the decode d may be changed, but the encode e will remain the same. But no two rooms can ever be opened by the same card.

3 Basic Notions

Let P and C be sets. We will call P the plaintext symbols, and C the codetext symbols. Let e be a subset of $P \times C$, called the encoding relation, and d a subset of $C \times P$, called the decoding relation. Then $\mathcal{R} = (P, C, e, d)$ is called a **precode**. $\mathcal{R}$ is said to be a **code** if and only if $d \circ e$ is a subdiagonal relation on P .

A **precode homomorphism** from (P, C, e, d) to (P', C', e', d') is defined by a list of functions $(h, k, h \times k, k \times h)$, where $h: P \rightarrow P'$ and $k: C \rightarrow C'$ are required to satisfy $(h \times k)(e) \subseteq e'$ and $(k \times h)(d) \subseteq d'$. Sometimes we will write $\langle h, k \rangle$ to denote this homomorphism.

We obtain a **category $\mathfrak{P}$ of precodes** by taking precodes as **objects** and precode homomorphisms as **morphisms**.

The identity functions on the plaintext and the codetext symbols of a precode $\mathcal{R}$ induce the **identity morphism** $1_{\mathcal{R}}$ associated to the precode $\mathcal{R}$. The composition of morphisms is given by the composition of functions. The class of objects in the category $\mathfrak{P}$ is denoted by $\text{Obj}(\mathfrak{P})$. The set of morphisms from $\mathcal{R}$ to $\mathcal{A}$ is denoted by $\text{Hom}_{\mathfrak{P}}(\mathcal{R}, \mathcal{A})$ or sometimes simply by $\mathcal{R} \rightarrow \mathcal{A}$.

Let $\mathcal{R} = (P, C, e, d)$ and $\mathcal{A} = (P', C', e', d')$ be precodes. $\mathcal{R}$ is said to be a **subprecode** of $\mathcal{A}$ if and only if $P \subseteq P'$, $C \subseteq C'$, $e \subseteq e'$, and $d \subseteq d'$. Notice that a subprecode of a code is again a code.

The **subcategory $\mathfrak{C}$ of codes** of the category $\mathfrak{P}$ of precodes is defined in the obvious way, taking codes as objects. Note that $\mathfrak{C}$ is a **full** subcategory of $\mathfrak{P}$. Indeed, consider a precode homomorphism between two codes $\mathcal{K}$ and $\mathcal{L}$. The image under this homomorphism is a subprecode of $\mathcal{L}$, and hence a subcode. Thus the set $\text{Hom}_{\mathfrak{P}}(\mathcal{K}, \mathcal{L})$ of precode morphisms coincides with the set of code morphisms $\text{Hom}_{\mathfrak{C}}(\mathcal{K}, \mathcal{L})$.

The code $\mathcal{I} = (\emptyset, \emptyset, \emptyset, \emptyset)$ is an **initial object** in the category of precodes, that is, there exists exactly one morphism from $\mathcal{I}$ to any another object $\mathcal{R}$ in $\mathfrak{P}$. Any code $\mathcal{T} = (\{p\}, \{c\}, \{(p, c)\}, \{(c, p)\})$ with singleton set symbols is a **terminal object** in $\mathfrak{P}$. The category $\mathfrak{P}$ does not have a zero object.

4 Morphisms

We take a closer look at the morphisms of precodes in this section. It turns out that the monomorphisms are just the injective functions respecting the encoding and decoding relations. We will see that epimorphisms need not be so well-behaved. Indeed, neither the category of precodes nor the category of codes is balanced, that is, bimorphisms need not be isomorphisms.

A morphism $f: \mathcal{A} \rightarrow \mathcal{B}$ of precodes $\mathcal{A}$ and $\mathcal{B}$ is said to be a **monomorphism**, or simply **monic**, if and only if $fg = fg'$ implies $g = g'$ for all $g, g' \in \text{Hom}_{\mathfrak{P}}(\mathcal{R}, \mathcal{A})$ and all $\mathcal{R} \in \text{Obj}(\mathfrak{P})$.

Lemma 1. *Let $f: \mathcal{R} \rightarrow \mathcal{A}$ be a morphism of precodes. Then $f = \langle f_1, f_2 \rangle$ is monic if and only if f_1 and f_2 are injective functions.*

Proof. Suppose that f_1 and f_2 are injective, hence monic, morphisms in the category of sets. This immediately implies that f is monic. Conversely, suppose that f is monic. Denote by $\mathcal{S} = (\{p\}, \{c\}, \emptyset, \emptyset)$ a precode with singleton symbol sets. Let x and y be (necessarily constant) morphisms from $\mathcal{S}$ to $\mathcal{R}$. Since $fx = fy$ implies $x = y$, it follows that f_1 and f_2 are injective. $\square$

A morphism $f: \mathcal{R} \rightarrow \mathcal{A}$ of precodes $\mathcal{R}$ and $\mathcal{A}$ is said to be an **epimorphism** if and only if $gf = g'f$ implies $g = g'$ for all $g, g' \in \text{Hom}_{\mathfrak{P}}(\mathcal{A}, \mathcal{B})$ and all $\mathcal{B} \in \text{Obj}(\mathfrak{P})$.

Lemma 2. *Let $f: \mathcal{R} \rightarrow \mathcal{A}$ be a morphism of precodes. Then $f = \langle f_1, f_2 \rangle$ is epic if and only if f_1 and f_2 are surjective functions.*

Proof. Suppose that f_1 and f_2 are surjective functions, hence epimorphisms, in the category of sets. This implies that f is an epimorphism.

Let f be an epimorphism. Seeking a contradiction, we assume that not both f_1 and f_2 are surjective. Denote by $\mathbf{2}$ the two element set $\{0, 1\}$. Define the precode $\mathcal{R} = (\mathbf{2}, \mathbf{2}, \mathbf{2} \times \mathbf{2}, \mathbf{2} \times \mathbf{2})$. Let g and h be two *distinct* morphisms in $\text{Hom}_{\mathfrak{P}}(\mathcal{A}, \mathcal{B})$ that take the same values on the image of f . It follows from our assumption that such morphisms exist. However, since $gf = hf$ implies $g = h$, we get the desired contradiction. $\square$

A morphism $f: \mathcal{R} \rightarrow \mathcal{A}$ of precodes $\mathcal{R}$ and $\mathcal{A}$ is called an **isomorphism** if and only if there exists a morphism $g: \mathcal{A} \rightarrow \mathcal{R}$ such that $fg = 1_{\mathcal{A}}$ and $gf = 1_{\mathcal{R}}$.

Lemma 3. *Let $f: \mathcal{R} \rightarrow \mathcal{A}$ be a morphism of precodes. If $f = \langle f_1, f_2 \rangle$ is an isomorphism then f_1 and f_2 are bijective functions.*

Proof. An isomorphism is monic and epic, implying that f_1 and f_2 are bijective functions. $\square$

The evident asymmetry in the statement of this lemma reflects the fact that an epimorphism $f = \langle f_1, f_2 \rangle$ need not be surjective on the encoding relation or decoding relation, even though the functions f_1 and f_2 are surjective. This fact has some quizzical consequences. For instance, a monic and epic morphism in the category of precodes is not necessarily an isomorphism. To see this, let ι denote the identity function on $\mathbf{2}$. Then $\langle \iota, \iota \rangle$ is a monic and epic morphism from $\mathcal{R} = (\mathbf{2}, \mathbf{2}, \emptyset, \emptyset)$ to $\mathcal{A} = (\mathbf{2}, \mathbf{2}, \mathbf{2} \times \mathbf{2}, \mathbf{2} \times \mathbf{2})$. But it is obviously not an isomorphism.

5 Limits

In this section we derive some fairly general constructions of codes and precodes. The constructions are based on the categorical notion of a limit.

Recall that a **diagram** D in a category $\mathfrak{P}$ is a directed graph whose vertices $i \in I$ are labelled by objects $\mathcal{R}_i$ in $\mathfrak{P}$ and whose edges $i \rightarrow j$ are labelled by morphisms in $\text{Hom}_{\mathfrak{P}}(\mathcal{R}_i, \mathcal{R}_j)$. The underlying graph is called the **scheme** of the diagram.

A family of morphisms $(f_i: \mathcal{A} \rightarrow \mathcal{R}_i)_{i \in I}$ with common domain $\mathcal{A}$ is said to be a **cone** for D , provided that for each arrow $d: \mathcal{R}_i \rightarrow \mathcal{R}_j$ in the diagram D , the triangle

$$\begin{array}{ccc} \mathcal{A} & & \\ f_i \downarrow & \searrow f_j & \\ \mathcal{R}_i & \xrightarrow{d} & \mathcal{R}_j \end{array}$$

commutes. A **limit** for D is a cone for D with the universal property that any other cone for D uniquely factors through it. In other words, if $(f_i: \mathcal{A} \rightarrow \mathcal{R}_i)_{i \in I}$ is the limit of a diagram D and $(g_i: \mathcal{B} \rightarrow \mathcal{R}_i)_{i \in I}$ is a cone for D , then there exists exactly one arrow $u: \mathcal{B} \rightarrow \mathcal{A}$ such that $g_i = f_i \circ u$ for all $i \in I$.

We want to show that the category $\mathfrak{P}$ of precodes and the category $\mathfrak{C}$ of codes are complete. In other words, we need to show that limits exist for all diagrams. Fortunately, it is sufficient to prove that products and equalizers exists [1,7].

We need to introduce some more notation. Let $(r_i)_{i \in I}$ be a family of relations indexed by a set I , where $r_i \subseteq P_i \times C_i$. We can define a product of these relations by

$$\prod r_i = \left\{ (p, c) \in \prod P_i \times \prod C_j \mid \forall i \in I (p(i), c(i)) \in r_i \right\},$$

where all products range over the index set I . Sometimes we will denote the product of two relations r_i and r_j by $r_i \otimes r_j$. For example, if $r_1 = \{(2, 1), (1, 2)\}$, $r_2 = \{(a, b), (a, c)\}$, then the product relation $r_1 \otimes r_2$ is given by

$$r_1 \otimes r_2 = \{((2, a), (1, b)), ((2, a), (1, c)), ((1, a), (2, b)), ((1, a), (2, c))\}.$$

Theorem 1. *The category $\mathfrak{P}$ of precodes has products. The product of a family of codes is again a code.*

Proof. Let $\mathcal{R}_i = (P_i, C_i, e_i, d_i)$, $i \in I$, be a family of precodes indexed by a set I . The product of this family is obtained by taking cartesian products of the symbol sets, and the product of the encoding and decoding relations. In other words, the product of the family $\mathcal{R}_i$ is given by $(\mathcal{R}, (\pi_i: \mathcal{R} \rightarrow \mathcal{R}_i)_{i \in I})$, where the precode $\mathcal{R}$ is given by the object $(\prod_{i \in I} P_i, \prod_{i \in I} C_i, \prod_{i \in I} e_i, \prod_{i \in I} d_i)$, and the projection map π_i is the obvious map onto the i th component. It is clear that $\mathcal{R}$ is a code if and only if all $\mathcal{R}_i$ are codes. $\square$

The equalizer $(\mathcal{E}, u)$ of two morphisms $f, g: \mathcal{R} \rightarrow \mathcal{A}$ is an object $\mathcal{E}$ together with a morphism $u: \mathcal{E} \rightarrow \mathcal{R}$ such that $fu = gu$, with the additional property that every morphism h satisfying $fh = gh$ factors uniquely through u . In other words, the triangle in the following diagram commutes:

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{u} & \mathcal{R} \xrightleftharpoons[g]{f} \mathcal{A} \\ \uparrow \epsilon & \nearrow h & \\ \mathcal{B} & & \end{array}$$

Recall that in the category of sets, the equalizer of two functions $f, g: R \rightarrow A$ is given by the coincidence set $\{x \in R \mid f(x) = g(x)\}$ with the inclusion mapping.

Theorem 2. *The category $\mathfrak{P}$ has equalizers. If $(\mathcal{E}, u)$ is the equalizer of two morphisms between codes, then $\mathcal{E}$ is also a code.*

Proof. Let $\mathcal{R} = (P, C, e, d)$ and $\mathcal{A}$ be precodes. Let $f = \langle f_1, f_2 \rangle$ and $g = \langle g_1, g_2 \rangle$ be a pair of morphisms between $\mathcal{R}$ and $\mathcal{A}$. We give an explicit construction of the equalizer.

The equalizer $(\mathcal{E}, u)$ of f and g is given by the precode $\mathcal{E} = (P^*, C^*, e^*, d^*)$, where the plaintext symbols $P^* = \{a \in P \mid f_1(a) = g_1(a)\}$ and codetext symbols $C^* = \{a \in C \mid f_2(a) = g_2(a)\}$ are just coincidence sets, and the encoding and decoding relations are obtained from $\mathcal{R}$ by restriction, that is, $e^* = e|_{P^* \times C^*}$, $d^* = d|_{C^* \times P^*}$, and the morphism $u = \langle \iota_1, \iota_2 \rangle$ is induced by the set inclusion maps $\iota_1: P^* \rightarrow P$, $\iota_2: C^* \rightarrow C$.

The construction ensures that $u(\mathcal{E})$ is the largest subprecode of $\mathcal{R}$ such that the restrictions of the functions f and g on $u(\mathcal{E})$ coincide, $f|_{u(\mathcal{E})} = g|_{u(\mathcal{E})}$. We can express h by a composition of a morphism $\epsilon: \mathcal{B} \rightarrow \mathcal{E}$ with u , since $h(\mathcal{B})$ is a subprecode of $u(\mathcal{E})$. The morphism ϵ is uniquely determined, since u is a monomorphism. $\square$

Theorem 3. *The category $\mathfrak{P}$ of precodes and the category $\mathfrak{C}$ of codes are complete.*

Proof. The categories $\mathfrak{P}$ and $\mathfrak{C}$ have products and equalizers and are therefore complete [1,7]. The main idea of this standard construction goes as follows. Suppose that we are given a diagram D in $\mathfrak{P}$ with sets V of vertices and E of edges. We build two products: the product of all objects in D , and the product indexed by E of all codomains of arrows in D . The universal property of the E -indexed product induces unique maps ψ_1 and ψ_2 as is shown in the following diagram:

$$\begin{array}{ccccc} & & & & \mathcal{R}_j \\ & & & \nearrow \pi_j & \uparrow \tilde{\pi}_e \\ \mathcal{L} & \xrightarrow{h} & \prod_{i \in V} \mathcal{R}_i & \xrightarrow[\psi_2]{\psi_1} & \prod (\mathcal{R}_j \mid i \xrightarrow{e} j \in E) \\ & \searrow c_i & \downarrow \pi_i & & \downarrow \tilde{\pi}_e \\ & & \mathcal{R}_i & \xrightarrow{d_e} & \mathcal{R}_j \end{array}$$

The map h is given by the equalizer of ψ_1 and ψ_2 , and the maps c_i are given by composition of h with the projection maps π_i , that is, $c_i = \pi_i h$. It is not difficult to see that $(\mathcal{L}, (c_i)_{i \in I})$ is a cone of D . It follows from the universality of the equalizer and of the V -indexed product that this cone is the limit of D . $\square$

6 Colimits

Reversing arrows, we obtain the concept of **cocones** and **colimits** of diagrams. We will derive the dual results for precodes.

Theorem 4. *The category $\mathfrak{P}$ of precodes has coproducts. The coproduct of a family of codes is again a code.*

Proof. The coproduct $(\mathcal{K}, (\iota_i: \mathcal{R}_i \rightarrow \mathcal{K})_{i \in I})$ of the family $\mathcal{R}_i$ is given by the disjoint union of the symbol sets and the induced disjoint union of the encoding and decoding relations together with the obvious inclusion maps. In other words,

$$\mathcal{K} = \left(\bigcup_{i \in I} P_i \times \{i\}, \bigcup_{i \in I} C_i \times \{i\}, \bigcup_{i \in I} e_i \otimes \Delta_i, \bigcup_{i \in I} d_i \otimes \Delta_i \right),$$

where Δ_i denotes the relation $\Delta_i = \{(i, i)\}$.

It is clear that $\mathcal{K}$ is a code if and only if all $\mathcal{R}_i$ are codes. $\square$

Theorem 5. *The category $\mathfrak{P}$ has coequalizers.*

Proof. Let $\mathcal{R}$ and $\mathcal{A} = (P, C, e, d)$ be precodes, and let $f = \langle f_1, f_2 \rangle$ and $g = \langle g_1, g_2 \rangle$ be a pair of morphisms between $\mathcal{R}$ and $\mathcal{A}$. Let E_1 be the smallest equivalence relation on P such that $f_1(a)$ and $g_1(a)$ are equivalent. Similarly, let E_2 be the smallest equivalence relation on C such that $f_2(a)$ and $g_2(a)$ are equivalent. The coequalizer of f and g is given by the precode $(P/E_1, C/E_2, e/E_1 \otimes E_2, d/E_2 \otimes E_1)$ and the morphism $\langle c_1, c_2 \rangle$ induced by the canonical quotient maps $c_1: P \rightarrow P/E_1$ and $c_2: C \rightarrow C/E_2$. $\square$

Remark 1. The coequalizer of two codes in the category $\mathfrak{P}$ is not necessarily a code. For example, let $\mathcal{K} = (\{1, 2\}, \{1, 2\}, id, id)$ the code with identity encoding and decoding relations. Denote by i and s the two bijective functions from $\{1, 2\}$ into itself. Then the coequalizer of the morphisms $\langle i, i \rangle$ and $\langle i, s \rangle$ is the precode $\mathcal{E}$ given by

$$\mathcal{E} = (\{1, 2\}, \{[1]\}, \{(1, [1]), (2, [1])\}, \{([1], 1), ([1], 2)\}).$$

Theorem 6. *The category $\mathfrak{P}$ of precodes is cocomplete.*

Proof. The category $\mathfrak{P}$ has coproducts and coequalizers and is therefore cocomplete. $\square$

7 Examples

Example 1 (RSA). Denote by p and q two distinct odd primes. A key setting of an RSA public key cryptosystem [10] can be seen as a code over the symbol set $\mathbf{Z}/pq\mathbf{Z}$, where the encoding relation e is given by the function $x \mapsto x^\varepsilon \bmod pq$ and the decoding relation d is given by $x \mapsto x^\delta \bmod pq$. The exponents are assumed to satisfy the congruence $\varepsilon\delta \equiv 1 \bmod \varphi(pq)$, where φ is Euler's totient function. We denote this code by $\mathcal{RSA} = (\mathbf{Z}/pq\mathbf{Z}, \mathbf{Z}/pq\mathbf{Z}, e, d)$.

Reducing the symbol sets modulo p and q respectively, one obtains two key-settings of Pohlig-Hellman cryptosystems [8], denoted by

$$\mathcal{PH}_1 = (\mathbf{Z}/q\mathbf{Z}, \mathbf{Z}/q\mathbf{Z}, e_1, d_1) \quad \text{and} \quad \mathcal{PH}_2 = (\mathbf{Z}/p\mathbf{Z}, \mathbf{Z}/p\mathbf{Z}, e_2, d_2).$$

The encoding and decoding relations are obtained from e and d by reducing modulo p and q respectively. For instance, the relation e_1 is given by the function $x \mapsto x^\varepsilon \bmod q$.

The $\mathcal{RSA}$ code is, in the terminology introduced in the next section, an example of a **product** of the codes $\mathcal{PH}_1$ and $\mathcal{PH}_2$.

Example 2 (RSA, cont'd). Conversely, given two Pohlig-Hellman codes

$$\begin{aligned} \mathcal{PH}_1 &= (\mathbf{Z}/q\mathbf{Z}, \mathbf{Z}/q\mathbf{Z}, x \mapsto x^{\varepsilon_1} \bmod q, x \mapsto x^{\delta_1} \bmod q) \\ \mathcal{PH}_2 &= (\mathbf{Z}/p\mathbf{Z}, \mathbf{Z}/p\mathbf{Z}, x \mapsto x^{\varepsilon_2} \bmod p, x \mapsto x^{\delta_2} \bmod p) \end{aligned}$$

and assuming that $\gcd(p-1, q-1) \mid (\varepsilon_1 - \varepsilon_2)$, then it is easy to see that the greatest common divisor of $p-1$ and $q-1$ divides $\delta_1 - \delta_2$. The Chinese remainder theorem yields the integers ε, δ satisfying

$$\begin{aligned} \varepsilon &\equiv \varepsilon_1 \bmod q-1, & \delta &\equiv \delta_1 \bmod q-1, \\ \varepsilon &\equiv \varepsilon_2 \bmod p-1, & \delta &\equiv \delta_2 \bmod p-1, \end{aligned}$$

respectively. The $\mathcal{RSA}$ code

$$(\mathbf{Z}/pq\mathbf{Z}, \mathbf{Z}/pq\mathbf{Z}, x \mapsto x^\varepsilon \bmod pq, x \mapsto x^\delta \bmod pq)$$

is then isomorphic to the product of $\mathcal{PH}_1$ and $\mathcal{PH}_2$.

Example 3 (Unequal Error Protection). We construct a simple (nonlinear) error control code that protects 0 and 1 against one single error, and can detect a single error in the transmission of 20 other symbols $\{2, \dots, 21\}$. This code is constructed with the help of two smaller codes.

Denote by $\mathbf{F}_2$ the binary finite field. Let C_1 be the set of all codewords in $\mathbf{F}_2^6$ of (Hamming) weight 0, 1, 5, and 6. Let $\mathcal{A}_1$ be the code $(\mathbf{F}_2, C_1, e_1, d_1)$, where $e_1(0) = 000000$ and $e_1(1) = 111111$, and the decoding relation d_1 maps all codewords of weight 0 or 1 to the plaintext symbol 0, and maps all codewords of weight 5 or 6 to the plaintext symbol 1.

Let C_2 be the set of all codewords in $\mathbf{F}_2^6$ of weight 3. The plaintext symbol set is given by $P_2 = \{2, \dots, 21\}$. The encoding relation e_2 maps the symbols $2, \dots, 21$ to the codewords in C_2 in lexical order respectively, and the decoding relation is given by the inverse function $d_2 = e_2^{-1}$. Then $\mathcal{A}_2 = (P_2, C_2, e, e^{-1})$.

The code $\mathcal{R}$ is given by the union of the codes $\mathcal{A}_1$ and $\mathcal{A}_2$, that is,

$$\mathcal{R} = (\{0, \dots, 22\}, \mathbf{F}_2^6, e_1 \cup e_2, d_1 \cup d_2).$$

This code is (isomorphic to) the **coproduct** (as defined in section 6) of the codes $\mathcal{A}_1$ and $\mathcal{A}_2$.

Example 4 (Codes over p-adic Integers). The famous explanation of the nonlinear Kerdoc and Preparata error control codes as linear codes over $\mathbf{Z}/4\mathbf{Z}$ gave rise to other explorations of Hensel lifting in coding theory. In [4], Calderbank and Sloane investigated a series of Hamming codes over the symbol sets $\mathbf{Z}/2^n\mathbf{Z}$. The familiar binary [7,4] Hamming code has generator polynomial $x^3 + x + 1$. Hensel lifting of this generator polynomial to $\mathbf{Z}/4\mathbf{Z}$ gives a unique monic irreducible polynomial that divides $x^7 - 1$ in $\mathbf{Z}/4\mathbf{Z}[x]$. Proceeding further, one obtains a series of cyclic codes over $\mathbf{Z}/8\mathbf{Z}$, $\mathbf{Z}/16\mathbf{Z}$, $\mathbf{Z}/32\mathbf{Z}$, etc. The 2-adic lift of the binary Hamming code is then the error control code over the ring of 2-adic integers with generator matrix

$$\begin{pmatrix} 1 & \lambda & \lambda^* & -1 & 0 & 0 & 0 \\ 0 & 1 & \lambda & \lambda^* & -1 & 0 & 0 \\ 0 & 0 & 1 & \lambda & \lambda^* & -1 & 0 \\ 0 & 0 & 0 & 1 & \lambda & \lambda^* & -1 \end{pmatrix},$$

where λ is the 2-adic integer $(1 - \sqrt{-7})/2$, and $\lambda^* = \lambda - 1$.

The code $(\mathbf{Z}_2^4, \mathbf{Z}_2^7, e, d)$ corresponding to this Hamming code over the 2-adic integers $\mathbf{Z}_2$ is a special case of the **limit** construction of codes described in Section 5.

8 Subprecode Lattice

The following question was posed in [3]: Does there exist a Jordan-Hölder-Schreier theory of codes? We give an affirmative answer to this question in this section.

Denote by $\text{Lat}(\mathcal{R})$ the set of subprecodes of a precode $\mathcal{R}$. The subprecode relation defines a partial order $\leq$ on $\text{Lat}(\mathcal{R})$, namely, $\mathcal{R}_i \leq \mathcal{R}_j$ if and only if $\mathcal{R}_i$ is a subprecode of $\mathcal{R}_j$.

Proposition 1. *The partially ordered set $\text{Lat}(\mathcal{R})$ of subprecodes of a precode $\mathcal{R}$ is a lattice. In particular, the subcodes of a code form a lattice.*

Proof. Define the **join** $\mathcal{R}_i \vee \mathcal{R}_j$ of two precodes $\mathcal{R}_i$ and $\mathcal{R}_j$ by their union $\mathcal{R}_i \vee \mathcal{R}_j = (P_i \cup P_j, C_i \cup C_j, e_i \cup e_j, d_i \cup d_j)$; and define the **meet** $\mathcal{R}_i \wedge \mathcal{R}_j$ of two precodes

$\mathcal{R}_i$ and $\mathcal{R}_j$ by their intersection $\mathcal{R}_i \wedge \mathcal{R}_j = (P_i \cap P_j, C_i \cap C_j, e_i \cap e_j, d_i \cap d_j)$. Clearly, $\mathcal{R}_i \vee \mathcal{R}_j$ is the smallest precode containing $\mathcal{R}_i$ and $\mathcal{R}_j$, and $\mathcal{R}_i \wedge \mathcal{R}_j$ is the largest subprecode contained in both $\mathcal{R}_i$ and $\mathcal{R}_j$. Thus, $\text{Lat}(\mathcal{R})$ is indeed a lattice with respect to those meet and join operations. The second statement follows immediately, since every subprecode of a code is again a code. $\square$

Notice that the lattice $\text{Lat}(\mathcal{R})$ is bounded, since all subprecodes $\mathcal{A}$ of $\mathcal{R}$ satisfy $0 \leq \mathcal{A} \leq 1$, where the bounds 0 and 1 are given by $0 = (\emptyset, \emptyset, \emptyset, \emptyset)$ and $1 = \mathcal{R}$. The lattice $\text{Lat}(\mathcal{R})$ is distributive, since the distributive laws of the meet and join operations

$$\begin{aligned} \forall \mathcal{A}, \mathcal{B}, \mathcal{C} \in \text{Lat}(\mathcal{R}): \quad \mathcal{A} \wedge (\mathcal{B} \vee \mathcal{C}) &= (\mathcal{A} \wedge \mathcal{B}) \vee (\mathcal{A} \wedge \mathcal{C}), \\ \forall \mathcal{A}, \mathcal{B}, \mathcal{C} \in \text{Lat}(\mathcal{R}): \quad \mathcal{A} \vee (\mathcal{B} \wedge \mathcal{C}) &= (\mathcal{A} \vee \mathcal{B}) \wedge (\mathcal{A} \vee \mathcal{C}), \end{aligned}$$

follow immediately from the set theoretic union and intersection properties. Thus, we can strengthen the statement of Proposition 1 as follows:

Proposition 2. *The partially ordered set $\text{Lat}(\mathcal{R})$ of subprecodes of a precode $\mathcal{R}$ is a bounded distributive lattice.*

A Schreier refinement theorem can be derived for any modular lattice, and thus in particular for the distributive lattice $\text{Lat}(\mathcal{R})$. We need to introduce some terminology to state this result. Let $\mathcal{A}$ and $\mathcal{B}$ be two precodes in $\text{Lat}(\mathcal{R})$ such that $\mathcal{A} \leq \mathcal{B}$. The subset $[\mathcal{A}, \mathcal{B}] = \{\mathcal{C} \in \text{Lat}(\mathcal{R}) \mid \mathcal{A} \leq \mathcal{C} \leq \mathcal{B}\}$ is called the **interval** between the precodes $\mathcal{A}$ and $\mathcal{B}$. Two **chains** in a subprecode lattice $\text{Lat}(\mathcal{R})$,

$$\mathcal{A} = \mathcal{A}_0 \leq \dots \leq \mathcal{A}_m = \mathcal{B}, \quad (1)$$

$$\mathcal{A} = \mathcal{B}_0 \leq \dots \leq \mathcal{B}_n = \mathcal{B}, \quad (2)$$

between the same subprecodes $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{R}$ are said to be **isomorphic** if and only if $m = n$ and there is a permutation π of $1, \dots, n$ such that the interval $[\mathcal{A}_{i-1}, \mathcal{A}_i]$ is lattice-isomorphic to the interval $[\mathcal{B}_{\pi(i)-1}, \mathcal{B}_{\pi(i)}]$. Defining the precodes $\mathcal{A}_{01} = \mathcal{B}_{01} = \mathcal{A}$, and

$$\mathcal{A}_{ij} = (\mathcal{A}_i \wedge \mathcal{B}_j) \vee \mathcal{B}_{j-1}, \quad \mathcal{B}_{ji} = (\mathcal{B}_j \wedge \mathcal{A}_i) \vee \mathcal{A}_{i-1},$$

for $i = 1, \dots, n$ and $j = 1, \dots, m$, we obtain a refinement of chain (1) and (2) by

$$\mathcal{A} = \mathcal{A}_{01} \leq \dots \leq \mathcal{A}_{m1} \leq \mathcal{A}_{12} \leq \dots \leq \mathcal{A}_{m2} \leq \mathcal{A}_{13} \leq \dots \mathcal{A}_{mn} = \mathcal{B},$$

$$\mathcal{A} = \mathcal{B}_{01} \leq \dots \leq \mathcal{B}_{n1} \leq \mathcal{B}_{12} \leq \dots \leq \mathcal{B}_{n2} \leq \mathcal{B}_{13} \leq \dots \mathcal{B}_{nm} = \mathcal{B},$$

respectively. Since the lattice $\text{Lat}(\mathcal{R})$ is modular, these two chains are isomorphic, cf. [5, p. 70]. Therefore, one obtains the following Schreier-type refinement proposition for precodes:

Proposition 3. *Any two chains between two precodes in $\text{Lat}(\mathcal{R})$ have isomorphic refinements.*

As a consequence, we obtain a Jordan-Hölder-type proposition:

Proposition 4. *Suppose that the precode $\mathcal{R} = (P, C, e, d)$ has finite symbol sets. Then any chain can be refined to a maximal chain and any two maximal chains between two given end-points have the same length.*

Unfortunately, this proposition is not as useful as its group theoretic analogue, since a maximal chain reflects the size of the precode and not its structure. For example, suppose that the cardinalities of P , C , e and d are α, β, γ and δ . Then all maximal chains are of length $\alpha + \beta + \gamma + \delta$.

9 Conclusions

A cryptanalyst who breaks a monoalphabetic substitution cipher by uncovering successively the plaintext value of various codetext symbols (e.g. by means of frequency analysis) does what amounts to forming an increasing sequence of subcodes of the cipher under attack. Similarly, an attack on a polyalphabetic substitution cipher which recovers one alphabet after another can be viewed as discovering homomorphic images of that cipher. It may often make sense to approach a cryptanalytic problem as a sequence of breaks of a sequence of homomorphic images of, or subobjects of, a code which is a key-setting of a cryptosystem.

The four propositions above form one schema for the first of these two approaches, but can involve lengthy maximal chains. A complementary – perhaps more incisive – Jordan-Hölder-Schreier theory might be obtained by recourse to a different partial order on a collection of precodes, such as an order based on homomorphic images (or even the strong homomorphic images suggested by the three isomorphism theorems in [3]).

This paper has shown that the general theory of codes introduced in [2,3] can be formulated in category-theoretical terms. It has presented constructions such as limits, colimits, equalizers, and has showed that special cases are in fact already present in existing codes in current use.

Acknowledgements. A.K. thanks the Santa Fe Institute for support through their Fellow-at-Large program.

References

1. M.A. Arbib and E. Manes. *Arrows, Structures, and Functors – The Categorical Imperative*. Academic Press, New York, 1975.
2. G.R. Blakley and I. Borosh. A general theory of codes, I: Basic concepts. In D. Dorninger, G. Eigenthaler, H.K. Kaiser, H. Kautschitsch, and W. More, editors, *Proceedings of the Klagenfurt Conference*, volume 10 of *Contributions to General Algebra*, pages 1–29. Verlag Johannes Heyn, Klagenfurt, Austria, 1998.

3. G.R. Blakley and I. Borosh. A general theory of codes, II: Paradigms and homomorphisms. In E. Okamoto, G. Davida, and M. Mambo, editors, *Information Security, First International Workshop, ISW '97*, volume 1396 of *LNCS*, pages 1–30. Springer Verlag, Berlin, 1998.
4. A.R. Calderbank and N.J.A. Sloane. Modular and p-adic cyclic codes. *Designs, Codes, and Cryptography*, 6:21–35, 1995.
5. P.M. Cohn. *Universal Algebra*. D. Reidel Publishing Company, Dordrecht, revised edition, 1981.
6. W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22:644–654, 1976.
7. S. Mac Lane. *Categories for the Working Mathematician*. Springer Verlag, Berlin, 1971.
8. S.C. Pohlig and M.E. Hellman. An improved algorithm for computing logarithms over $\text{GF}(p)$ and its cryptographic significance. *IEEE Trans. Inform. Theory*, 24:106–110, 1978.
9. G.P. Purdy. A high-security log-in procedure. *Comm. of the ACM*, 17(4):442–445, 1974.
10. R.L. Rivest, A. Shamir, and Adleman L.M. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978.

Cryptanalysis of Polynomial Authentication and Signature Scheme

Hongjun Wu, Feng Bao, Dingfeng Ye, and Robert H. Deng

Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613

{hongjun, baofeng, dfye, deng}@krdl.org.sg

Abstract. Polynomial Authentication and Signature Scheme (PASS) is a new public key authentication and signature scheme proposed by NTRU Cryptosystems Inc. It is based on the hard problems related to constrained polynomial evaluation. In this paper, we break PASS with the proposed parameters. We show how to forge valid authentication transcripts or digital signatures in PASS with knowledge of the public key only and without knowing any previous authentication transcripts or signatures.

1 Introduction

A group of highly efficient public key cryptosystems, including NTRU [1, 2] for asymmetrical encryption and PASS [3] for digital authentication and signature, were proposed recently by Hoffstein, Pipher, and Silverman from NTRU Cryptosystem, Inc. This group of cryptosystems is based on the hard problems of partial evaluation of constrained polynomial over polynomial rings. An outstanding feature of the proposed cryptosystems is that key generation, encryption/decryption, authentication and digital signature operations can be performed highly efficiently.

NTRU, after its first appearance at Crypto'96, has received a great deal of attentions in the cryptographic community. Odlyzko first pointed out the meet-in-the-middle attack on NTRU and this was followed by the lattice attacks on NTRU from Coppersmith and Shamir [4]. The reaction attack proposed recently by Hall, Goldberg and Schneier [5] also has effect on NTRU. The current suggested parameters for NTRU and corresponding security estimates take into account all the above attacks. For a discussion on security of NTRU with respect all these attacks, the reader is referred to the NTRU Cryptosystem's homepage (<http://www.ntru.com/documentcenter.htm>).

The NTRU cryptosystem has also received a lot of attentions from industry. Sony Corporation of America has made an investment in NTRU, and Tao Group Ltd of Reading, England, signed an agreement with NTRU which will make the incorporation of encryption technologies into handheld and consumer devices (<http://www.ntru.com/pressreleases/>). The NTRU public key encryption scheme was submitted to IEEE P1363 group as a potential IEEE public key cryptosystem standard (<http://grouper.ieee.org/groups/1363/addendum.html>).

More recently, Hoffstein, Pipher, and Silverman from NTRU Cryptosystem Inc proposed PASS, a public key cryptosystem for authentication and digital signature [3]. Like NTRU, PASS features extremely light computational requirements for both the prover and the verifier. The hard problem underlying the security of PASS is related to properties of short polynomials. Since short polynomials can be made to correspond to short vectors in a lattice, it is important to carefully consider the possibility of attack by lattice reduction methods in the design of PASS. The authors of PASS were well aware of lattice reduction methods such as the LLL [6] and the improved LLL lattice reduction methods [7, 8] and they designed PASS to be secure against these attacks.

In this paper, we present the cryptanalysis of PASS. We show that PASS with the proposed parameters is not sound, i. e., it is easy to forge one's authentication transcripts or signatures knowing only the public key and without any knowledge of the private key or any previous authentication transcripts or signatures. The amount of computation required to launch the attacks is small enough to be carried out very comfortably on a PC.

This paper is organized as follows. Section 2 presents the PASS authentication and signature scheme. Two attacks to PASS authentication and signature schemes are given in Section 3 and Section 4, respectively. Section 5 concludes the paper.

2 Description of PASS

Define a ring of truncated polynomials as

$$R = (Z/qZ)[x]/(x^N - 1) \quad (1)$$

where q is a prime number and N is a divisor of $q-1$. A typical element g of R is denoted as a polynomial or a vector

$$g = g_0 + g_1x + g_2x^2 + \cdots + g_{N-1}x^{N-1} = [g_0 \ g_1 \ g_2 \ \cdots \ g_{N-1}]$$

where $g_i \in Z/qZ$. The value of $g(\alpha)$ is computed as $g(\alpha) \bmod q$ for every $\alpha \in Z/qZ$. The norm of g is a real number given by

$$\|g\| = \sqrt{r_0^2 + r_1^2 + r_2^2 + \cdots + r_{N-1}^2},$$

where

$$r_i = \begin{cases} g_i & \text{if } g_i < q/2 \\ q - g_i & \text{if } g_i > q/2 \end{cases}.$$

A polynomial is called short if its norm is small. Since multiplication in R is $\bmod x^N - 1$, it is equivalent to a cyclic convolution product. For example, the product of f and g is given by

$$h = f \cdot g \text{ with } h_k = \sum_{i+j \equiv k \bmod N} f_i g_j.$$

Since the multiplication of $h = f \cdot g$ is $\bmod x^N - 1$, it follows that

$$h(\alpha) = f(\alpha) \cdot g(\alpha) \bmod q$$

for every α satisfying $\alpha^N \equiv 1 \bmod q$.

A set S of t distinct non-zero elements $\alpha \in \mathbb{Z}/q\mathbb{Z}$ is chosen as a system wide parameter. Each element α of S is chosen such that $\alpha^N \equiv 1 \bmod q$ and $\alpha^{-1} \in S$. Also public are four subsets of R , denoted as L_f , L_g , L_c and L_h , respectively, and defined as follows. Fix a positive integer $d_f < N/2$. Define L_f to be the set of all polynomials f in R such that f has d_f coefficients equal to each of 1 and -1, with all other coefficients equal to 0. The norm of f is thus $\sqrt{2d_f}$. And L_g and L_c are defined similarly using d_g and d_c . L_h is defined as the set of polynomials h in R satisfying $|h| < \gamma_h q$ for a specified γ_h .

PASS is based on the fact that the product of two extremely short polynomials is still a short polynomial.

2.1 PASS Authentication Scheme

Alice, the prover, has a private key consisting of two polynomials f and f' that are chosen randomly from L_f . Her public key is the values of $f(\alpha)$ and $f'(\alpha)$ for all $\alpha \in S$, where S is the public system parameter given above. To prove to Bob, the verifier, that she possesses the secret key f , f' associated to her public key, Alice proceeds as follows:

- Alice randomly chooses g, g' from L_g and reveals the values of $g(\alpha)$ and $g'(\alpha)$ for all $\alpha \in S$. This is Alice's commitment.
- Bob randomly chooses a challenge c_0 and sends c_0 to Alice. c_0 is hashed with the commitment to produce polynomials: $c_1, c_2, c_3, c_4 \in L_c$.
- Alice reveals the polynomial $h = c_1 f g + c_2 f g' + c_3 f' g + c_4 f' g'$.
- Bob verifies that
 - 1) $h \in L_h$ (i.e., $|h| < \gamma_h q$)
 - 2) $h(\alpha) = c_1(\alpha) f(\alpha) g(\alpha) + c_2(\alpha) f(\alpha) g'(\alpha) + c_3(\alpha) f'(\alpha) g(\alpha) + c_4(\alpha) f'(\alpha) g'(\alpha)$
 for all the $\alpha \in S$.

2.2 PASS Signature Scheme

The private key and the public key in the signature scheme are the same as that in the authentication scheme. Alice, the signer, is to sign a message M with her private key. She proceeds as follows:

- Alice randomly chooses the polynomials g, g' from L_g and computes $g(S)$ (i. e., $\{g(\alpha) \mid \alpha \in S\}$) and $g'(S)$.
- Alice hashes M with $g(S)$ and $g'(S)$ to construct polynomials: $c_1, c_2, c_3, c_4 \in L_c$.
- Alice computes h as

$$h = c_1fg + c_2fg' + c_3f'g + c_4f'g'$$

- The signature is $(g(S), g'(S), h)$.
- The verification process in the signature scheme is the same as that in the authentication scheme, except that $c_1, c_2, c_3, c_4 \in L_c$ are generated from $g(S), g'(S)$, and M instead of the challenge.

2.3 A Specific Example of PASS

The following parameters are suggested for PASS in [3]:

$$\begin{aligned} q &= 769, \quad N = 768, \quad t = N/2 = 384 \\ d_f &= 256, \quad d_g = 256, \quad d_c = 1, \quad \gamma_h = 2.2 \end{aligned}$$

where N is simply chosen as $q-1$. It is estimated in [3] that PASS with such suggested parameters would be more secure than 1024 bit RSA since it takes longer time to recover the PASS private key. Unfortunately, as we will demonstrate in the remaining part of the paper that PASS with these parameters is extremely weak and can be broken with small amount of computations.

3 Cryptanalysis of the PASS Authentication Scheme

It was conjectured in [3] that for large N and t slightly larger than $N/2$, PASS would be sound. However, such conjecture is not true. We will present an attack in this section to forge the authentication transcript independent of the sizes of N and t .

In PASS, recovering the private key f and f' from the public key is expected to be difficult. Moreover, for polynomials f and f' chosen randomly from L_f , given

$f(\alpha)$ and $f'(\alpha)$ for all $\alpha \in S$, it is difficult to find short polynomials (with small norms) f_x and f'_x to satisfy $f_x(\alpha) = f(\alpha)$ and $f'_x(\alpha) = f'(\alpha)$ for all $\alpha \in S$. However, we notice that in order to satisfy

$$h(\alpha) = c_1(\alpha)f(\alpha)g(\alpha) + c_2(\alpha)f(\alpha)g'(\alpha) + c_3(\alpha)f'(\alpha)g(\alpha) + c_4(\alpha)f'(\alpha)g'(\alpha)$$

for all $\alpha \in S$ (i.e., the second condition verified by the verifier Bob), it is not necessary for f_x and f'_x to satisfy $f_x(\alpha) = f(\alpha)$ and $f'_x(\alpha) = f'(\alpha)$ for all $\alpha \in S$. The reason is that if $g(\alpha) = g'(\alpha) = 0$ for a particular $\alpha \in S$, then $f_x(\alpha)$ and $f'_x(\alpha)$ can be set to any arbitrary values at that α . If we can construct short polynomials g and g' such that $g(\alpha) = g'(\alpha) = 0$ for most of the $\alpha \in S$, then the short polynomials f_x and f'_x only need to satisfy $f_x(\alpha) = f(\alpha)$ and $f'_x(\alpha) = f'(\alpha)$ for a few $\alpha \in S$ and they can be constructed easily. In the following, we show how to generate the short polynomials g and g' with $g(\alpha) = g'(\alpha) = 0$ for most of the $\alpha \in S$.

Theorem 1. Let R be the ring defined in (1), q be a prime number, $N = q - 1$ and p be a divisor of N . If $g \in R$ and the coefficients of g are with period p , i.e., $g_i = g_{(i+kp) \bmod N}$ for any value of k , then there are at most p non-zero elements of Z/Zq satisfying $g(\alpha) \neq 0$.

Proof: The polynomial g with period p can be denoted as

$$g(x) = (g_0 + g_1x + g_2x^2 + \cdots + g_{p-1}x^{p-1})(1 + x^p + x^{2p} + \cdots + x^{N/p}).$$

For $(x^p - 1) \bmod p \neq 0$, $g(x)$ can be written as

$$g(x) = (g_0 + g_1x + g_2x^2 + \cdots + g_{p-1}x^{p-1}) \frac{x^N - 1}{x^p - 1}.$$

Note that $\alpha^N - 1 \equiv 0 \bmod q$, thus the necessary condition for $g(\alpha) \neq 0$ is that $\alpha^p - 1 \equiv 0 \bmod q$. Since there are at most p distinct solutions in Z/Zq for $\alpha^p - 1 \equiv 0 \bmod q$, there are at most p non-zero elements in Z/Zq satisfying $g(\alpha) \neq 0$.

According to theorem 1, the short polynomials g and g' with $g(\alpha) = g'(\alpha) = 0$ for most of the $\alpha \in S$ can be constructed easily if N has small factors. We simply choose their coefficients from the set $\{-1, 0, 1\}$ to generate short polynomials and with short period. Since the proposed value of N is $768 = 3 \times 2^8$, there are a number of small factors of N for g and g' to be constructed easily. For example, if we set the coefficients of short polynomials g and g' with period 6, then most likely there will be about $(768 - 6)/2 = 381$ values of $\alpha \in S$ with $g(\alpha) = g'(\alpha) = 0$ since the elements of S are chosen in [3] without being aware of this attack.

After obtaining the desired short polynomials g and g' , we then proceed to construct the short polynomials f_x and f'_x . Denote S_1 as the subset of S containing all those $\alpha \in S$ with $g(\alpha) = g'(\alpha) = 0$ and denote $S_2 = S - S_1$. Now, short polynomials f_x and f'_x are only required to satisfy $f_x(\alpha) = f(\alpha)$ and $f'_x(\alpha) = f'(\alpha)$ for all $\alpha \in S_2$. The construction of f_x and f'_x becomes much easier especially when the number of elements in S_2 is sufficiently small. For example, if there are only 3 elements in S_2 , even an exhaustive search (by randomly choosing the coefficients from the set $\{-1, 0, 1\}$) can be applied to determine the short polynomials f_x and f'_x since only about $769^3 = 2^{28.8}$ trials are needed in this scenario. It should be noted that such computations are done prior to the authentication process. With the four short polynomials g, g', f_x and f'_x , valid authentication transcripts can be produced. This attack is valid as long as N has many small factors, like 768, regardless of the size of the challenge.

To summarize the result in this section, the PASS authentication scheme with the proposed parameters is not secure, i.e. it is not sound. A cheater, who knows only Alice's public key, can produce valid authentication transcripts on Alice's behalf easily without knowing Alice's private key. The amount of computation required is small and no previous authentication transcripts are needed in the attack.

4 Cryptanalysis of the PASS Signature Scheme

The attack described in Section 3 can be applied directly to break the PASS signature scheme.

The attack in Section 3 depends essentially on the fact that $N = 768$ has small factors. It is thus necessary to choose N as a prime number instead of $q-1$. In this section, another attack is applied to the PASS signature scheme even if N is chosen as a prime number. In this attack, the small space of L_c would enable signatures being forged easily.

We notice that the h can be written as

$$\begin{aligned} h &= c_1 f g + c_2 f g' + c_3 f' g + c_4 f' g' \\ &= c_1 (f + r_1 f') g + c_2 (f + r_2 f') g' \\ &= c_1 h_1 + c_2 h_2 \end{aligned}$$

where $c_3 = c_1 \cdot r_1$, $c_4 = c_2 \cdot r_2$, $h_1 = (f + r_1 f') g$ and $h_2 = (f + r_2 f') g'$. We could specify two polynomials r_1 and r_2 (the simplest way is to set $r_1 = r_2 = 1$). We then choose arbitrarily two short polynomials h_1 and h_2 . Another two polynomials g and g' are computed to satisfy $h_1(\alpha) = (f(\alpha) + r_1(\alpha) f'(\alpha)) g(\alpha)$ and $h_2(\alpha) = (f(\alpha) + r_2(\alpha) f'(\alpha)) g'(\alpha)$ for all the $\alpha \in S$. Here g and g' are not required to be short polynomials. After obtaining g and g' , we compute c_1, c_2, c_3 and c_4 . If

$c_3 = c_1 \cdot r_1$ and $c_4 = c_2 \cdot r_2$ with the pre-specified polynomials r_1 and r_2 , we forged successfully a signature $(g(S), g'(S), h)$ in which

$$h = c_1 h_1 + c_2 h_2$$

The probability of success of one trial is about $1/|L_c|^2$. Repeat this attack by choosing different polynomials h_1 and h_2 , the signature could finally be forged with about $0.5 \times |L_c|^2$ trials, which is about $2^{-37.3}$ for $N = 768$ and $d_c = 1$.

The detailed attack is given in the rest of this section. Let f and f' be Alice's private key and $f(\alpha)$ and $f'(\alpha)$ be the corresponding public key for all $\alpha \in S = \{\alpha_1, \alpha_2, \dots, \alpha_t\}$. The attack is as follows.

1. Arbitrarily choose $r_1, r_2 \in R$ such that for any $c \in L_c$, $r \cdot c \in L_c$.
2. Compute $\beta_{1i} = f(\alpha_i) + r_1(\alpha_i)f'(\alpha_i)$ and $\beta_{2i} = f(\alpha_i) + r_2(\alpha_i)f'(\alpha_i)$ for $i = 1, 2, \dots, t$. Assume that $\beta_{1i} \neq 0, \beta_{2i} \neq 0$ for all $i = 1, 2, \dots, t$ (we will deal with the case when some of them are 0 later).
3. Let F_1, F_2 be two functions such that $F_1(\alpha_i) = \beta_{1i}, F_2(\alpha_i) = \beta_{2i}$ for $i = 1, 2, \dots, t$. Choose two arbitrary short polynomials h_1, h_2 . Compute the polynomials G_1, G_2 satisfying $h_1 = F_1 G_1, h_2 = F_2 G_2$ on S .
4. Hash message M with $G_1(S), G_2(S)$ to generate c_1, c_2, c_3 and c_4 . If the generated c_1, c_2, c_3 and c_4 happen to satisfy $c_3 = r_1 c_1$ and $c_4 = r_2 c_2$, we set $h = c_1 h_1 + c_2 h_2$ and $g = G_1, g' = G_2$. Otherwise, go to step 3 and repeat the attack.
5. $(h, g(S), g'(S))$ is a valid signature of M .

In step 4, we note that the generated c_1, c_2, c_3 and c_4 satisfy $c_3 = r_1 c_1$ and $c_4 = r_2 c_2$ with probability $1/|L_c|^2$. That is the success rate for one trial.

In this attack, step 1 and step 4 can be carried out easily as shown in the Theorem 2 and 3 in the Appendix, respectively.

In step 2 we assume that $\beta_{1i} \neq 0, \beta_{2i} \neq 0$ for all $i = 1, 2, \dots, t$. Such assumption is, however, not always true. In the following, we improve the attack so that it works even if up to four elements of $\{\beta_{1i} | i = 1, 2, \dots, t\}$ and four elements of $\{\beta_{2i} | i = 1, 2, \dots, t\}$ are zero. We start with the following two facts.

Fact 1. Let $c_1, c_2 \in L_c$. If each of h_1, h_2 has $N - 4$ small coefficients and four arbitrary coefficients, $h = c_1 h_1 + c_2 h_2$ is with norm $|h| < 2.2q$.

It is because that

- 1) h has $N - 16$ small coefficients and 16 arbitrary coefficients;

- 2) For the small coefficients, the sum of h_i^2 is less than $0.8q^2$ if $h_i < 0.8\sqrt{q}$. This is fulfilled if each of the small coefficients of h_1, h_2 is smaller than $0.2\sqrt{q}$.
- 3) From the definition of norm in Section 2, $h_i^2 < q^2/4$ for any h_i .
- 4) Thus $\sum_{i=0}^{N-1} h_i^2 < 4q^2 + 0.8q^2$. It implies $|h| < 2.2q$;

Fact 2. Let $F \in R$ and $F(\alpha) = 0$ for up to 4 $\alpha \in S \subset Z/qZ - \{0\}$. Based on Theorem 4 in the Appendix, it is easy to construct a $G \in R$, such that H ($H = FG$) has $N - 4$ arbitrarily small coefficients.

With Fact 1 and Fact 2, our attack can be applied in the situation where up to four elements of $\{\beta_{1i} \mid i=1,2,\dots,t\}$ and four elements of $\{\beta_{2i} \mid i=1,2,\dots,t\}$ are zero. For $N = 768$ and $t = N/2$, the probability that more than four elements of $\{\beta_{1i} \mid i=1,2,\dots,t\}$ or more than four elements of $\{\beta_{2i} \mid i=1,2,\dots,t\}$ are 0 is only about 2^{-12} . So the improved attack would succeed without being significantly affected by the values of β_{1i} and β_{2i} .

Similar attack could be applied to the PASS authentication scheme. The valid authentication transcript can be generated with probability $1/|L_c|^2$. This value is about $2^{-38.3}$ for the proposed parameters. The PASS authentication scheme with the proposed parameters is thus not secure with respect to this attack.

5 Conclusion

PASS is a highly efficient public key authentication and signature scheme that was designed to resist LLL lattice reduction method and the improved LLL method. The authors of PASS estimated that the breaking time for the proposed parameter $N = 768$ is approximately 4.73×10^{18} MIP-years [3]. In this paper we showed that the above claim is false and that PASS with the proposed parameter is not sound. That is, one can forge authentication transcripts or signatures without recovering the private key and with small amount of computations.

We presented two attacks to PASS. Both attacks apply to the PASS authentication scheme and the PASS signature scheme, though we only demonstrated the first attack to the PASS authentication scheme in Section 3 and the second attack to the PASS signature scheme in Section 4. The first attack succeeded in forging authentication transcripts (or signatures) by exploiting the fact that the proposed parameter $N = 768$ has small factors. To resist this attack, we suggest choosing N as a prime number. However, a prime N can not resist the second attack presented in Section 4, which succeeded in breaking PASS using the fact that the space L_c is too small.

Enlarging q is not an effective counter measure against our second attack since q must be very large in order to give PASS a reasonable security. In that case, its computational efficiency will degrade significantly. Modifications may be applied to other parameters of PASS such as d_f, d_g, d_c , etc; however, their security implications must be considered very carefully since the corresponding γ_h must be changed in that case. And γ_h is a key factor for the security of PASS under other attacks such as LLL attack.

Generally, we believe that any further modification to PASS should take the attacks presented in this paper into consideration.

References

1. J. Hoffstein, J. Pipher and J.H. Silverman, „NTRU: A New High Speed Public Key Cryptosystem“, preprint; presented at the rump session of *Crypto'96*.
2. J. Hoffstein, J. Pipher and J.H. Silverman, „NTRU: A Ring Based Public Key System“, *Proceedings of ANTS III*, Porland (1998), Springer-Verlag.
3. J. Hoffstein, D. Lieman and J.H. Silverman, „Polynomial Rings and Efficient Public Key Authentication“, *Proceedings of International Workshop on Cryptographic Techniques and E-Commerce*, pp 7-19, Ed M. Blum and C. H. Lee, July 5-8, 1999, Hong Kong. Also available at <http://www.ntru.com/documentcenter.htm>.
4. D. Coppersmith and A. Shamir, „Lattice Attacks on NTRU, Preprint, April 5, 1997“, presented at *Eurocrypt'97*.
5. C. Hall, I. Goldberg and B. Schneier, „Reaction Attacks Against Several Public-Key Cryptosystems“, preprint April 1999, available at <http://www.counterpane.com>.
6. A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovsz, „Factoring Polynomials with Rational Coefficients“, *Mathematische Ann.*, 261 (1982), pp. 513-634.
7. C. P. Schnorr, „A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms“, *Proceedings of Theoretical Computer Science*, 53 (1987), pp. 201-224.
8. C. P. Schnorr, „A More Efficient Algorithm for Lattice Basis Reduction“, *J. Algorithms*, 9 (1988), pp. 47-62.

Appendix

Theorem 2. There exists a polynomial $r \in R = (Z/qZ)[x]/(x^N - 1)$ such that for any $c \in L_c$, the product $r \cdot c \in L_c$.

Proof: The validity of this theorem is trivial by noting that x^i fulfills the requirement for any i .

Theorem 3. Let $F \in R$ and $F(\alpha) \neq 0$ for $\alpha \in S \subset Z/qZ - \{0\}$. For any $H \in R$, it is easy to construct a $G \in R$, such that $F(\alpha)G(\alpha) = H(\alpha)$ for all $\alpha \in S$.

Proof: Given H , the values of $H(\alpha)$ for all non-zero $\alpha \in S$ can be computed. Define $G(\alpha) = H(\alpha)/F(\alpha)$ for $\alpha \in S$ and arbitrarily set $G(\alpha)$ for $\alpha \in Z/qZ - \{0\} - S$. The coefficients of G can be computed directly since

$$\begin{pmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{N-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_N & \cdots & \alpha_N^{N-1} \end{pmatrix}$$

is a non-singular matrix for different $\alpha_1, \alpha_2, \dots, \alpha_N$.

Denote $F = a_0 + a_1x + \cdots + a_{N-1}x^{N-1}$ and matrix $\mathbf{F}$ as

$$\mathbf{F} = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} \\ a_{N-1} & a_0 & a_1 & \cdots & a_{N-2} \\ a_{N-2} & a_{N-1} & a_0 & \cdots & a_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{pmatrix}$$

Lemma 1. The rank of $\mathbf{F}$ equals to the amount of $\alpha \in Z/qZ - \{0\}$ satisfying $F(\alpha) \neq 0$.

Proof. Let $E = b_0 + b_1x + \cdots + b_{N-1}x^{N-1}$ and $FE = c_0 + c_1x + \cdots + c_{N-1}x^{N-1}$, we must have

$$\begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} \\ a_{N-1} & a_0 & a_1 & \cdots & a_{N-2} \\ a_{N-2} & a_{N-1} & a_0 & \cdots & a_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{pmatrix} \begin{pmatrix} b_{N-1} \\ b_{N-2} \\ b_{N-3} \\ \vdots \\ b_0 \end{pmatrix} = \begin{pmatrix} c_{N-1} \\ c_{N-2} \\ c_{N-3} \\ \vdots \\ c_0 \end{pmatrix}$$

Consider all the E satisfying $FE = 0$. All these E (their coefficient vectors) consist of a linear space. It is the null space of $\mathbf{F}$. Every such E must satisfy $E(\alpha) = 0$ for those α such that $F(\alpha) \neq 0$. By counting the number of such E we reach Lemma 1.

Theorem 4. Let $F \in R$ and $F(\alpha) = 0$ for exactly k $\alpha \in Z/qZ - \{0\}$. It is easy to construct a $G \in R$ so that the polynomial $H = FG$ is with $N - k$ arbitrarily small coefficients.

Proof. From Lemma 1 the rank of $\mathbf{F}$ is $N - k$. There are k linear independent rows in $\mathbf{F}$. Without loss of generality, assume that the first $N - k$ rows are linear independent. Arbitrarily set the last $N - k$ coefficients of H as small numbers $h_{N-1}, h_{N-2}, \dots, h_k$. Solve G from

$$\left(\begin{array}{c} \text{the first } N - k \\ \text{rows of } \mathbf{F} \end{array} \right) \begin{pmatrix} g_{N-1} \\ g_{N-2} \\ g_{N-3} \\ \vdots \\ g_0 \end{pmatrix} = \begin{pmatrix} h_{N-1} \\ h_{N-2} \\ \vdots \\ h_k \end{pmatrix}$$

Secure Transactions with Mobile Agents in Hostile Environments

Panayiotis Kotzanikolaou¹, Mike Burmester², and Vassilios Chrissikopoulos¹

¹University of Piraeus, Department of Informatics,
Karaoli & Dimitriou 80, 185 34 Piraeus, Greece
{pkotzani, chris}@unipi.gr

²Information Security Group, Royal Holloway,
University of London, Egham, Surrey TW20 OEX, UK
m.burmester@rhbnc.ac.uk

Abstract. A major problem of mobile agents is their apparent inability to authenticate transactions in hostile environments. In this paper, we consider a framework for the prevention of agent tampering without compromising the mobility or autonomy of the agent. Our approach uses encrypted functions. We present an RSA implementation which answers affirmatively the open problem on undetachable signatures of Sander and Tschudin.

1 Introduction

Traditional software programs use primarily the Remote Procedure Call (RPC) for remote applications, which is performed in accordance with the client – server architecture. The principle of Remote Programming (RP) forms an alternative to RPC. In the RP approach, the client object, instead of exchanging request and reply messages remotely, is itself transferred to the server and is executed locally. The mobile agent paradigm constitutes a way of implementing Remote Programming.

Mobile agents are autonomous software entities that are able to migrate across different execution environments. Mobility and autonomy make permanent connections unnecessary; thus mobile agents are suitable for providing low-bandwidth connections and asynchronous communication [1,2,9,21]. Furthermore, they provide better support for heterogeneous environments.

The characteristics of mobile agents make them ideal for electronic commerce applications in open networks. A mobile agent can search for special products or services and negotiate on behalf of its owner with other entities. Furthermore, mobile agents can be used as selling agents. However, mobile agents are vulnerable to several attacks and in particular to attacks by malicious hosts. Until quite recently there was a general belief that mobile agent vulnerability could be prevented only with hardware solutions. Chess *et al* state [2]: "It is impossible to prevent agent tampering unless trusted (and tamper-resistant) hardware is available... Without such hardware, a mali-

cious [host] can always modify/manipulate the agent". This belief however has been shown to be misleading and is referred to „*the Chess paradox*“ in the literature [15].

Jakobsson and Juels propose executable code (X-cash) which binds the user to a payment transaction [8]. The user links the executable code ω with a so-called negotiable certificate that warrants a restricted purchase. This protocol is practical but not very flexible, in the sense that it requires the issuing of a large number of negotiable certificates for certain purchases.

Sander and Tschudin [15] propose the use of encrypted functions. The user encrypts a function s , which is then executed by the host, without the host having access to s . Although the approach is very promising, no secure implementation has been proposed as yet. Our approach is also based on encrypted functions but is provably secure.

Organization of the Paper. In Section 2 we review the security issues of mobile agents. In Section 3 we discuss the requirements for secure computations with encrypted functions and review *undetachable* signatures. In Section 4 we consider a realization of an undetachable RSA signature scheme which allows a mobile agent to conduct a transaction inside a hostile host without being abused. This realization is provably secure and answers affirmatively the open problem of Sander and Tschudin on undetachable signatures [15]. We conclude in Section 5.

2 An Overview of the Security Issues for Mobile Agents

Although the mobile agent paradigm extends the capabilities of traditional ways of remote communication and distributed computing, it also raises new security issues [3]. These are generally divided into two broad areas: i) protecting the host from malicious agents, and ii) protecting the agent from hostile hosts.

Protecting the host from attacks by malicious agents is possible by using effective access control and sandbox mechanisms (e.g. Java's sandbox security component). A more challenging problem is to protect an agent from being abused by a hostile server. During the execution of a mobile agent, the agent is in a very asymmetric relation with regards to the server, since the server must be able to access the agent's code, data and state, in order to execute it. It is not clear how private information (such as a secret key) can be used by an agent, without revealing it to its executing environment. A hostile host could easily appropriate the secret key when the agent attempts to use it. It is believed that this is a very hard problem to solve [1]. For example, Yi *et al* [20] state: "Current consensus is that it is computationally impossible to protect mobile agents from malicious hosts. Instead of tackling the problem from a computational (difficult) point of view, current research is looking at sociological means of enforcing good host behavior".

Research efforts for solving the problem of hostile hosts are broadly divided into two main categories [15,17]:

- *Detection* of agent tampering.

This category includes solutions that aim to detect agent tampering *a posteriori*, trace the identity of the illegitimate host and prove its misbehavior. Vigna [6] introduced a tracing mechanism, which records the execution of the agent and its interaction with the execution environment. The tracing mechanism will reveal the malicious host. Yi *et al* proposed the use of an Agent Service Center [20], which traces the itinerary of the agent. Kotzanikolaou *et al* [9] use a multi-agent system that can trace which mobile agents were victims of malicious behavior.

These systems may provide partial solutions to particular problems. However, there are cases where *a posteriori* detection is not sufficient.

- *Prevention of agent tampering*

The philosophy of this approach is to prevent agent tampering *a priori*. There are two cases to be considered: *passive* and *active* prevention. *Passive* prevention mechanisms protect the agents by using organizational or architectural solutions. Farmer *et al* [5] propose a scheme where mobile agents are only able to circulate in trusted execution environments. Merwe and Sholms [12] introduce a trade agent system where the agents are implemented with distributed objects that communicate remotely. Some detection mechanisms also use passive prevention techniques, *e.g.* [9,20].

These approaches either make strong assumptions on the trustworthiness of a host [5], or compromise many of the advantages of mobile agents such as autonomy [9,20] and migration [12].

Active prevention focuses on the development of solutions that provide an agent with protection from attacks of hostile hosts, without compromising the advantages of the mobile agent paradigm. These may use secure hardware devices, *e.g.* [13,18]. However, their deployment is low, mainly because of the high costs involved. The exploration of active prevention mechanisms that are software-based, is a recent field of research. Fig.1 shows the security issues related to the mobile agent paradigm.

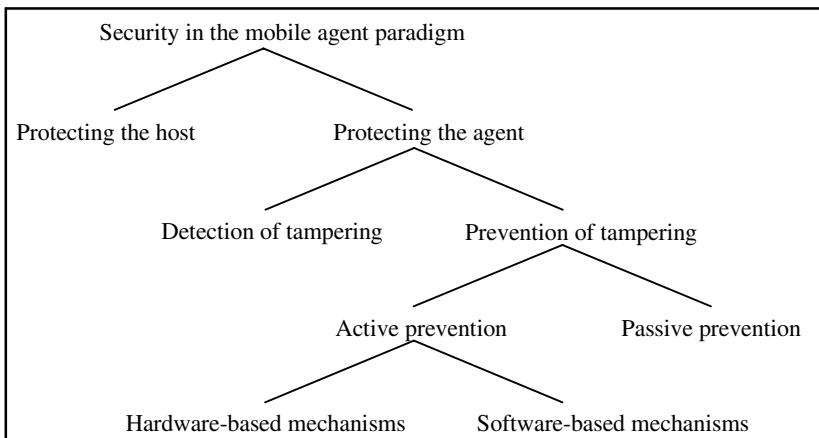


Fig. 1. Security issues in the mobile agent paradigm

A first approach for software-based active prevention of agent tampering is the use of obfuscation techniques or scrambling and mess-up algorithms, as proposed by Hohl

[6]. This approach could be useful in certain occasions where an agent carries time-limited token-data. However, the security of this method cannot be proven.

In [11] it is proposed a method which allows Alice to send mobile code in an untrusted host (Bob) and evaluate an encrypted Boolean function, while maintaining the confidentiality of the function. The security of the scheme is reduced to the security of the McEliece public key cryptosystem. Unfortunately, under this scheme it is not possible for anyone except Alice to decrypt the result, because Alice's private key is involved in the decryption process.

Sander and Tschudin [15] use a technique called Computing with Encrypted Functions (CEF). The host executes an encrypted function $s \circ f$ where f is the encrypting function, without having access to the function s . The security of the method relies on the difficulty of decomposing the encrypted function. Because the spirit of mobile agents is to perform computations in a mostly autonomous way, the authors in [15] explore the requirements for deploying non-interactive CEF. A candidate class of functions, which can be used for non-interactive CEF consists of the multivariate rational functions.

The authors in [15] observe that it would be useful to employ *algebraic* homomorphic functions in the CEF approach. Unfortunately, so far there are no known provably secure algebraic homomorphic encryption functions [15]. So the problem of prevention of agent tampering in hostile environments remains open.

3 Encrypting a Signature Function

Undetachable signatures were proposed by Sander and Tschudin and are based on non-interactive CEF [15].

Suppose that a Customer wishes to send a mobile Agent to purchase some goods from an electronic shop over the Internet. The Agent can autonomously authenticate the transaction only if it is able to use a signature function s of the Customer. However, the Agent is executed by a potentially hostile Server. To protect the signature function s , the Customer encrypts it with a function f to obtain

$$f_{\text{signed}} := s \circ f \quad (1)$$

and gives the pair of functions $(f(\cdot), f_{\text{signed}}(\cdot))$ to the Agent, as part of its executable code. On migration the Server executes the pair $(f(\cdot), f_{\text{signed}}(\cdot))$ on input x to obtain the *undetachable* signature pair

$$f(x) = m \quad \text{and} \quad f_{\text{signed}}(x) = s(f(x)) = s(m).$$

The pair of functions $(f(\cdot), f_{\text{signed}}(\cdot))$ enables the Agent to create signatures of the Customer on messages of the Server, without revealing the signature function s (the secret key of s) to the latter. The input x of the Server is linked to the Server's bid. The parameters of function f are such that the output of f includes the Customer's constraints. Thus, $m = f(x)$ links the constraints of the Customer to the bid of the Server. This is certified by $f_{\text{signed}}(x) = s(m)$ which employs the signature function s of the

Customer. The Server cannot use the pair (f, f_{signed}) to sign *arbitrary* messages, because the function f is linked to the constraints of the Customer. These constraints may include detailed description of the required product, the maximum acceptable price, time-stamps and other requirements of the Customer.

Requirements for Secure Undetachable Signatures

- It must be feasible for the Agent to execute the encrypted function (1) (in polynomial time) on the input x of the Server.
- The pair of functions f and s must be such that it is hard to get s by decomposing the encrypted function (1).

In the following section we give an implementation of undetachable signatures with RSA [14] signatures.

4 A Secure Solution for the Sander - Tschudin Open Problem on Undetachable Signatures

For a realization of undetachable signatures, the authors in [15] proposed among others, the use of birational functions as introduced by Shamir [16]. However, the schemes resulting from these constructions are insecure [15] and subject to the Coppersmith, Stern and Vaudenay attack [4]. The existence of a secure undetachable signature scheme remains an open problem [7,15].

Below we will describe a non-interactive CEF undetachable signature scheme, which uses exponential functions as encrypting functions instead of birational functions and is provably secure. Our protocol is based on the RSA cryptosystem.

A Protocol for Secure Transactions with Mobile Agents

Setting. We use an RSA setting. Each Customer selects a modulus n which is the product of two large primes p, q and a number $e, 1 < e < \varphi(n) = (p - 1)(q - 1)$, such that $\gcd(e, \varphi(n)) = 1$. Let d be such that $1 < d < \varphi(n) = (p - 1)(q - 1)$ and $de = 1 \bmod \varphi(n)$, and let *hash* be an appropriate hash function (e.g. MD5).

Let C be an identifier for the Customer, req_C the constraints of the Customer and $h = hash(C, req\_C)$ a binary string whose value is bounded by n . The constraints req_C define the requirements of the Customer for a specific purchase. These may include the description of a desired product, the maximum price that is acceptable by the Customer and a deadline for the delivery of the product. Furthermore, it may contain an expiration date and a timestamp. Furthermore, let S be an identifier for the Server and bid_S the bid of the Server, whose structure is analog to req_C .

Preparing the Agent. The Customer gives to the Agent as part of its executable code the undetachable signature function pair

$$f(.) = h^{(.)} \bmod n$$

and

$$f_{signed}(.) = k^{(.)} \bmod n$$

where $k = h^d \bmod n$ is the Customer's RSA signature of h . Observe that f_{signed} is the encryption *sof* of the RSA signature function $s(.) = (.)^d \bmod n$ of the Customer, that is:

$$f_{signed}(.) = s \circ f(.) = s(f(.)) = s(h^{(.)}) = (h^{(.)})^d = (h^d)^{(.)} = k^{(.)}.$$

The Agent then migrates to the Server with the pair of functions $(f(.), f_{signed}(.))$ as part of its code and with (C, req_C) as part of its data.

Executing the Agent. The Server executes the Agent on input $x = \text{hash}(S, C, bid\_S)$ to obtain the RSA signature (m, z) , with

$$m = f(x) = h^x \bmod n \text{ and}$$

$$z = f_{signed}(x) = k^x \bmod n = (h^d)^x \bmod n = (h^x)^d \bmod n = m^d \bmod n = s(m).$$

Parameters	Inputs
d : Customer's secret key n, e : Customer's public key req_C : The Customer's requirements (constraints)	bid_S : The Server's bid

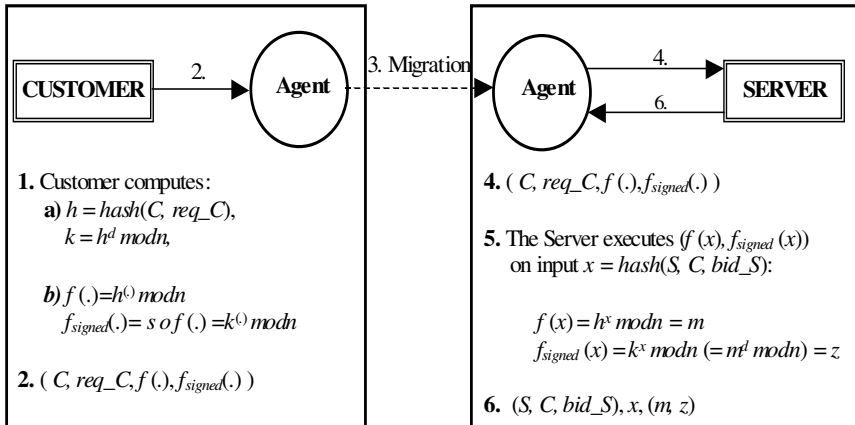


Fig. 2. A secure undetachable signature scheme based on RSA and exponential functions

In this protocol, the Agent is given the certified requirements of the Customer (h, k) , where $f(.) = h^{(.)} \bmod n$, $f_{signed}(.) = k^{(.)} \bmod n$. The Server modifies these by including the bid, bid_S in the input x , in such a way so as to get an undetachable signature (m, z)

for the transaction, where $m = f(x)$ and $z = f_{signed}(x)$. This serves as a certificate which is authenticated by the Customer ($z = m^d \bmod n$). The certified constraints of the Customer, req_C , and the bid of the Server, bid_S restrict the scope of the certificate (m, z) to "optimal bid" transactions with appropriate time-limits (or more generally, to whatever requirements the Customer and the Server stipulate). The protocol is illustrated in Fig.2. Note that even if a Server ignores the Customer's constraints req_C and executes the Agent (functions (f, f_{signed})) in order to produce an undetachable signature of the Customer for a bogus bid, the signature will be invalid. If a Server is not willing to bid for a purchase, then it forwards the Agent to an other Server.

Proposition. The undetachable signature scheme described above is as secure the RSA cryptosystem.

Proof. A hostile Server must successfully forge (with non-negligible probability) a new valid undetachable signature (h, k) for a transaction (x, m, z) , given a history of earlier valid transactions. Note that the certificate (h, k) is an integral part of the transaction. Clearly, there is no problem constructing (x, m, z) for a given (h, k) . The problem for a malicious Server is to construct a new RSA signature (h', k') of the Customer which will include modified constraints req_C' of the Customer. If this is possible, then one can also break the RSA signature scheme.

Main Features of the Scheme:

- *Efficiency:* The RSA implementation of our scheme involves only three exponentiations.
- The Server is able to bind the Customer to the transaction via an undetachable signature, which links the constraints of the Customer req_C and the bid of the Server bid_S .
- A malicious Server can produce a signature $s(m^*)$ of a message m^* that includes a bogus bid bid_S (non-compatible with req_C), but such a signature will be invalid.
- This protocol is asymmetric. The Server is not committed to the transaction, whereas the Customer is. The protocol is therefore subject to impersonation attacks on the Server. For symmetry the Customer may request that the Server sign the transaction to bind itself to the bid. In this application this is of little concern, because bogus bids bid_S will be discarded by the Customer. However, for applications in which the bid R_S must be authenticated, the Server is required to sign R_S .
- In the protocol described above there is no privacy. Exchanged data can be read by any eavesdropper. To achieve privacy, the Customer must encrypt the code and the data of the Agent with the public encryption key of the Server and the Server must encrypt the result of the Agent's execution with the public encryption key of the Customer.

- In order to avoid the possibility that an Agent binds the Customer to more than one valid bid of several Servers for a single purchase, a priority policy among bids must be set. In a simple solution, the Agent terminates when it finds the first acceptable bid. In case of dispute, time-stamps can be used and the Server that provides the oldest valid undetachable signature of the Customer is the winner. More sophisticated solutions could be constructed with the involvement of an Agent Broker. In such a scheme, the Agent Broker could act as a middleware between Customers and Servers of electronic shops and help in the achievement of optimal contracts.

5 Conclusion

Current research on software-based active prevention contradicts the general belief that mobile code vulnerability could be actively prevented only with hardware-based solutions. In this paper we prove that it is possible for mobile agents to conduct private and binding transactions in a hostile environment, by using cryptographic primitives.

We present a solution to the hostile host problem based on the RSA signature scheme, which is provably secure. This answers positively the Sander - Tschudin problem of undetachable signatures. Our basic scheme is asymmetric and does not bind the Server to the transaction. For a symmetric scheme we require that the Server also authenticate the transaction.

References

1. Chess, D., Harrison, C. and Kershenbaum, A.: Mobile Agents: Are They a Good Idea?, J. Vitek and C. Tschudin (ed.) Mobile Object Systems, Springer, (1996)
2. Chess, D., Grosz, B., Harrison, C., Levine, D., Parris, C. and Tsodik, G.: Itinerant Agents for Mobile Computing, Technical Report, RC 20010 (1995) IBM T.J. Watson Research Center, NY
3. Chess David M.: Security Issues in Mobile Code Systems. Mobile Agents and Security, LNCS Vol. 1419, Springer-Verlag (1998) 1–14
4. Coppersmith Don, Stern Jacques and Vaudenay Serge: Attacks on the Birational Permutation Signature Schemes. Douglas R. Stinson ed., Proceedings of Crypto'93, LNCS 773, Springer-Verlag (1993) 435–443
5. Farmer William, Gutmann Joshua and Swarup Vipin: Security for Mobile Agents: Authentication and State Appraisal. Proceedings of the European Symposium on Research in Computer Security (ESORICS), LNCS 1146, Springer-Verlag (1996) 118–130
6. Hohl Fritz: Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts. Mobile Agent Security, LNCS Vol. 1419, Springer-Verlag (1998) 92–113
7. Hohl, Fritz: A Model of Attacks of Malicious Hosts Against Mobile Agents. Proceedings of the ECOOP Workshop on Distributed Object Security and 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, INRIA, France, (1998) 105 - 120
8. Jakobsson Markus, Juels Ari: X-cash: Executable Digital Cash, LNCS Vol. 1465, Springer-Verlag (1998) 16 - 27

9. Kotzanikolaou Panayiotis, Katsirelos George and Chrissikopoulos Vassilios: Mobile Agents for Secure Electronic Transactions. Recent Advances in Signal Processing and Communications, World Scientific and Engineering Society Press (1999) 363–368
10. Lipton Richard and Sander Tomas: An Additively Homomorphic Encryption Scheme or How to Introduce a Partial Trapdoor in the Discrete Log, November (1997). Submitted for Publication
11. Loureiro Sergio, Molva Refik: Privacy for Mobile Code, In proceedings of Distributed Object Security Workshop OOPSLA'99, Denver, November 1999
12. Merwe, J. and Solms, S.H.: Electronic Commerce with Secure Intelligent Trade Agents. Proceedings of ICICS'97, LNCS Vol. 1334, Springer-Verlag (1997) 452–462
13. Palmer, E.: An Introduction to Citadel – a Secure Crypto Coprocessor for Workstations. Proceedings of the IFIP SEC'94 Conference, Curacao (1994)
14. Rivest Ronald, Shamir Adi and Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of ACM, Vol. 21 (2) (1978) 120–126
15. Sander Tomas and Tschudin Christian F.: Protecting Mobile Agents Against Malicious Hosts. Mobile Agent Security, LNCS Vol. 1419, Springer-Verlag, (1998) 44–60
16. Shamir Adi: Efficient Signature Schemes Based On Birational Premutations. Proceedings of the DARPA Workshop on foundations for secure mobile code, Monterey CA, USA, (1997)
17. Vigna Giovanni: Cryptographic Traces for Mobile Agents. Mobile Agent Security, LNCS Vol. 1419, Springer-Verlag, (1998) 137–153
18. Wilhelm, U.G.: Cryptographically Protected Objects. Technical Report, Ecole Polytechnique Federale de Lausanne, Switzerland (1997)
19. Yee Bennet S.: A Sanctuary for Mobile Agents. Proceedings of the DARPA workshop on foundations for secure mobile code, Monterey CA, USA (1997)
20. Yi Xun, Wang Xiao Feng and Lam Kwok Yan: A Secure Intelligent Trade Agent System. Proceedings of the International IFIP/GI Working Conference, TREC'98, LNCS Vol. 1402, Springer-Verlag (1998) 218–228
21. Zapf Michael, Muller Helge and Geihs Kurt: Security Requirements for Mobile Agents in Electronic Markets. Proceedings of the International IFIP/GI Working Conference, TREC'98, LNCS Vol. 1402, Springer-Verlag (1998) 205–217.

A Multisignature Scheme with Message Flexibility, Order Flexibility and Order Verifiability

Shirow Mitomi and Atsuko Miyaji

School of Information Science, Japan Advanced Institute of Science and Technology
miyaji@jaist.ac.jp

Abstract. Multisignature scheme realizes that plural users generate the signature on a message, and that the signature is verified. Various studies on multisignature have been proposed([4,13,11,8,1]). They are classified into two types: RSA([9])-based multisignature([4,8]), and discrete logarithm problem(DLP) based multisignature([13,11,1]), all of which assume that a message is fixed beforehand. In a sense, these protocols do not have a feature of message flexibility. Furthermore all schemes which satisfy with order verifiability designate order of signers beforehand [13, 1]. Therefore these protocols have a feature of order verifiability but not order flexibility.

For a practical purpose of circulating messages soundly through Internet, a multisignature scheme with message flexibility, order flexibility and order verifiability should be required. However, unfortunately, all previous multisignature do not realize these features. In this paper, we propose a multisignature scheme with flexibility and verifiability. We also present two practical schemes based on DLP based message recover signature([7]) and RSA signature([4]), respectively.

1 Introduction

In proportion as the spread of personal computers and network, messages like documents, data, software, etc., have been circulated through Internet. In such environment, an entity sends/forwards an original message to others, or sends a modified message to others. Through the process of circulation, a message has been improved or added a convenient feature one by one, and finally has been completed. However recently it has been a new problem for computer virus to be mixed into a message through the process of this circulation. Apparently it is an obstacle to circulate messages soundly through Internet. Another problem concerns the copyright: it is necessary to distinguish an original author from authors who modify an original message in a circulating message. This is why a multisignature scheme suitable for such an environment should be required.

Up to the present, various studies on multisignature have been proposed([4, 13,11,8,1]). They are classified into two types: RSA([9]) based multisignature([4, 8]), and discrete logarithm problem(DLP) based multisignature([13,11,1]). All

schemes assume that a message is fixed beforehand since they suppose the following scenario: a message fixed beforehand is passed and signed one by one through members in an organization like a company. Therefore these schemes cannot handle the following situation: an original message is passed and modified by unspecified entities. Furthermore we want to guarantee such circulating message in the next point: who writes an original message, who modifies the message, to which the message is modified, and how order the message is modified. In previous multisignature schemes([4,13,11,8,1]), signing from the first signer is obliged to start only if one of signers wants to modify a message: these do not have a feature of *message flexibility*. Furthermore [4,11,8] have a feature of *order verifiability* neither. Order verifiability is first realized in [13,1]. However they must designate order of signs beforehand. If we want to change order of signers, add a new signer, or exclude a signer, we are obliged to reset some data like public keys [1]: these have a feature of order verifiability but not *order flexibility*. Therefore previous schemes are not suitable for handling the above situation that a message circulates through unspecified entities.

In this paper, we propose a basic model of multisignature scheme that has the following three features:

Message flexibility: A message does not need to be fixed beforehand. Therefore each signer can modify an original message.

Order flexibility: Neither order of signers nor signers themselves need to be designated beforehand. Therefore we can easily change order of signers, add a new signer and exclude a signer.

Message and order verifiability: Each entity can verify who is an original author of a message, who modifies an original message and furthermore to which or how order a message is modified.

We also present two practical schemes based on the DLP based message recovery signature([7]) and RSA signature([4]). Furthermore we discuss some typical attacks against our scheme like a ordinary forgery, swapping order of signers, excluding a signer. We denote the functions to break DLP, forge our scheme in ordinary assumption, that in swapping order of signers, and that in excluding a signer, by DLP, FORGE, SWAP, and Exclude, respectively. Then we prove the following theorems by using polynomial-time truth-table($\leq_{tt}^{fp}$) reducibility of function:

(1) $\text{Forge} \equiv_{tt}^{fp} \text{DLP}$, (2) $\text{SWAP} \equiv_{tt}^{fp} \text{DLP}$, and (3) $\text{Exclude} \equiv_{tt}^{fp} \text{DLP}$.

Furthermore we investigate a feature of *Robustness* in a multisignature scheme: a message cannot be recovered if the signature verification fails. Because unauthentic message might damage a receiver especially in case that a message circulate through unspecified entities. Therefore the following feature should be required:

Robustness: If the signature verification on a message fails, then prevent such an unauthentic message from damaging a receiver.

We also propose a multisignature scheme with Robustness, *multisigncrypt*, which combines our multisignature with a function of encryption. Our multisigncrypt has a feature that a message cannot be recovered if the signature verification fails.

This paper is organized as follows. Section 2 summarizes a multisignature scheme([1]) and discusses several drawbacks in case that a message circulate through unspecified entities. Section 3 investigates a model of multisignature with flexibility and verifiability. Section 4 presents two practical schemes concretely and discusses the performance. Section 5 discusses the security on our multisignature scheme. Section 6 presents our multisigncrypt scheme.

2 Previous Work

In this section, we summarize a previous multisignature scheme([1]).

2.1 Previous Multisignature Scheme

We assume that n signers $I_1, I_2, \dots, I_n$ generate a signature on a fixed message M according to order fixed beforehand.

Initialization: A trusted center generates a prime p , $g \in \mathbb{Z}_p^*$ with prime order q , and set a hash function $h()$. A signer I_i generates a random number $a_i \in \mathbb{Z}_q^*$ as I_i 's secret key. Then I_i 's public key is computed sequentially as follows: $y_1 = g^{a_1} \pmod{p}$, $y_i = (y_{i-1} \cdot g)^{a_i} \pmod{p}$. Then a public key of ordered group $(I_1, I_2, \dots, I_n)$ is set to $y = y_n$.

Signature generation:

(1) Generation of r : Signer $I_1, \dots, I_n$ generate r together as follows.

1. I_1 selects $k_1 \in \mathbb{Z}_q^*$ randomly and computes $r_1 = g^{k_1} \pmod{p}$. If $\gcd(r_1, q) \neq 1$, then select new k_1 again.
2. For $i \in \{2, \dots, n\}$; a signer I_{i-1} sends r_{i-1} to I_i . I_i selects $k_i \in \mathbb{Z}_q^*$ randomly and computes $r_i = r_{i-1}^{a_i} \cdot g^{k_i} \pmod{p}$. If $\gcd(r_i, q) \neq 1$, then select new k_i again.
3. $r = r_i$.

(2) Generation of s : Signer $I_1, \dots, I_n$ generate s together as follows.

1. I_1 computes $s_1 = a_1 + k_1 r \cdot h(r, M) \pmod{q}$.
2. For $i \in \{2, \dots, n\}$; I_{i-1} sends s_{i-1} to I_i . I_i verifies that $g^{s_{i-1}} \stackrel{?}{=} y_{i-1} r_{i-1}^{r \cdot h(r, M)} \pmod{p}$, then computes $s_i = (s_{i-1} + 1)a_i + k_i r \cdot h(r, M) \pmod{q}$.
3. $s = s_i$.

(3) The multisignature on M by order $(I_1, \dots, I_n)$ is given by (r, s) .

Signature Verification: A multisignature (r, s) on M is verified by checking $g^s \stackrel{?}{=} y \cdot r^{r \cdot h(r, M)} \pmod{p}$.

2.2 Drawbacks

In this section, we discuss the drawbacks of the previous scheme in the following situation: each entity sends an original message or a modified message to others. In such a situation, a multisignature scheme should satisfy the following conditions:

Message flexibility: A message does not need to be fixed beforehand. Therefore each signer can modify an original message.

Order flexibility: Neither order of signers nor signers themselves need to be designated beforehand. Therefore we can easily change order of signers, add a new signer and exclude a signer.

Message and order verifiability: Each entity can verify who is an original author of a message, who modifies an original message and furthermore to which or how order a message is modified.

The previous multisignature has the following drawbacks considering the above situation although it realizes order flexibility:

1. A message M should be fixed beforehand. This scheme does not allow any signer to generate a signature on his modified message.
2. A public key for multisignature should be determined by order of signers. Therefore after setting up a public key for multisignature, a signer can be neither added nor excluded. Even order of signers cannot be changed.
3. The signature generation phase runs two rounds through all signers.

3 Our Basic Multisignature Scheme

This section proposes a basic model of multisignature schemes with flexibility and verifiability for both message and order. First we define the following notations. An original message M_1 is given by I_1 . $M_{1,2,...,i}$ ($i > 2$) denotes a message which is added some modification by the i -th signer I_i . The difference between $M_{1,2,...,i-1}$ and $M_{1,2,...,i}$, which means the modification by I_i , is defined as,

$$m_i = Diff(M_{1,2,...,i-1}, M_{1,2,...,i}).$$

We also define a function *Patch* which recovers a message,

$$M_{1,2,...,i} = Patch(m_1, m_2, ..., m_i).$$

For the sake of convenience, we denoted $m_1 = Patch(M_1)$. We use a signature scheme with message recovery feature. The signature generation or message recovery function is denoted by $Sign(sk_i, m_i) = sgn_i$, or $Rec(pk_i, sgn_i)$, respectively, where sk_i is I_i 's secret key and pk_i is I_i 's public key. Let h_1 be a hash function. We also use two operations $\otimes$ and $\odot$ in a group G

$$(A \otimes B) \odot B = A \ (\forall A, B \in G).$$

For example in case of $G = \mathbb{Z}_p$, $\otimes$ and $\odot$ mean modular multiplication and modular inversion, respectively. Then the signature generation and verification are done as follows. Figure 1 and 2 show the signature generation and verification, respectively.

Signature generation:

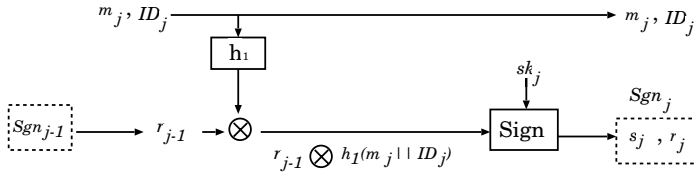


Fig. 1. I_j 's signature generation

1. The first signer I_1 generates a signature on $h_1(m_1 || ID_1)$ as follows,

$$sgn_1 = \text{Sign}(sk_1, h_1(m_1 || ID_1)) = (r_1, s_1),$$

where a signature sgn_1 is divided into two parts, r_1 and s_1 : r_1 is the next input to I_2 's signature generation, which is recovered by I_2 's signature verification. On the other hand, s_1 is the rest of sgn_1 , which is sent to all signers as it is. Then send (ID_1, s_1, r_1, m_1) as a signature on m_1 to the next.

2. A signer I_j receives messages $m_1, m_2, \dots, m_{j-1}$ from I_{j-1} . If $j > 2$, patch a message $M_{1,2,\dots,j-1}$ as follows,

$$M_{1,2,\dots,j-1} = \text{Patch}(m_1, m_2, \dots, m_{j-1}).$$

I_j modifies $M_{1,2,\dots,j-1}$ to $M_{1,2,\dots,j-1,j}$, computes the modification m_j ,

$$m_j = \text{Diff}(M_{1,2,\dots,j-1}, M_{1,2,\dots,j}),$$

and generates a signature on m_j by using r_{j-1} of I_j 's signature,

$$sgn_j = \text{Sign}(sk_j, r_{j-1} \otimes h_1(m_j || ID_j)) = (r_j, s_j),$$

where sgn_j is divided into r_j and s_j in the same way as the above. Then I_j 's signature on m_j is (r_j, s_j) .

3. A multisignature of $M_{1,2,\dots,i} = \text{Patch}(m_1, m_2, \dots, m_i)$ by $I_1, I_2, \dots, I_{i-1}$ and I_i is given by $(ID_1, s_1, m_1), (ID_2, s_2, m_2), \dots, (ID_i, s_i, r_i, m_i)$.

Signature verification:

1. A verifier receives $(ID_1, s_1, m_1), (ID_2, s_2, m_2), \dots, (ID_i, s_i, r_i, m_i)$ from a signer I_i .

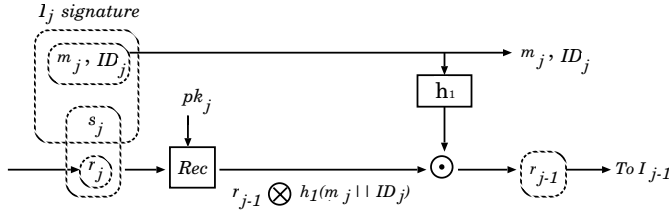


Fig. 2. I_j 's signature verification step

2. For $j = i, i - 1, \dots, 2$; compute

$$\begin{aligned} T_j &= \text{Rec}(pk_j, (r_j, s_j)) = r_{j-1} \otimes h_1(m_j || ID_j), \\ r_{j-1} &= T_j \odot h_1(m_j || ID_j). \end{aligned}$$

Let $j = j - 1$ and repeat step 2.

3. Finally compute

$$T_1 = \text{Rec}(PK_{p1}, (r_1, s_1)),$$

and verifies

$$T_1 \stackrel{?}{=} h_1(m_1 || ID_1)$$

Our basic model satisfies the three features, message flexibility, order flexibility, message verifiability and order verifiability. Furthermore, we easily see that any message recovery signature can be applied to the above basic model. In the next section, we present two schemes based on DLP and RSA.

4 Two Concrete Multisignature Schemes

In this section, we give two examples based on DLP and RSA.

4.1 DLP Based Scheme

There are many variants of DLP based schemes in both types of message with appendix([3,12,2]) and message recovery signature([6,7]). For the sake of convenience, here we use the message recovery signature with DSA-signature equation([7]). Apparently any message recovery signature scheme can be applied to our multisignature scheme.

Initialization: An authenticated center generates a large prime p , $g \in \mathbb{Z}_p^*$ with prime order q . Two $\mathbb{Z}_p$ -operations $\otimes$ and $\odot$ in section 3 are defined as multiplication and inverse in $\mathbb{Z}_p$, respectively. Each signer generates a pair of secret key $x_i \in \mathbb{Z}_q^*$ and a public key $y_i = g^{x_i} \pmod{p}$, and publish a public key y_i with his identity information ID_i .

Signature generation:

1. The first signer I_1 generates a signature on an original message m_1 . First generate $k_1 \in \mathbb{Z}_q$ randomly, compute $R_1 = g^{k_1} \pmod{p}$, $r_1 = (h_1(m_1 || ID_1))^{-1} \cdot R_1 \pmod{q}$, and $s_1 = (x_1 r_1 + 1) k_1^{-1} \pmod{q}$, where I_1 's signature on m_1 is (r_1, s_1) , and send (ID_1, s_1, r_1, m_1) to the next signer I_2 .
2. A signer $I_j (j \geq 2)$ receives $M_{1,2,\dots,j-1} = Patch(m_1, m_2, \dots, m_{j-1})$, modifies $M_{1,\dots,j-1}$ to $M_{1,\dots,j}$. Then I_j generates a signature on the difference $m_j = Diff(M_{1,\dots,j-1}, M_{1,\dots,j})$: generate $k_j \in \mathbb{Z}_q$ randomly, and compute $R_j = g^{k_j} \pmod{p}$, $r_j = (h_1(m_j || ID_j) \times r_{j-1})^{-1} \cdot R_j \pmod{q}$, and $s_j = (x_j r_j + 1) k_j^{-1} \pmod{q}$, where I_j 's signature on m_j is (r_j, s_j) .
3. A multisignature of $M_{1,2,\dots,i} = Patch(m_1, m_2, \dots, m_i)$ by $I_1, \dots, I_{i-1}$ and I_i is given by $(ID_1, s_1, m_1), \dots, (ID_{i-1}, s_{i-1}, m_{i-1}), (ID_i, s_i, r_i, m_i)$.

Signature verification

1. A verifier receives $(ID_1, s_1, m_1), \dots, (ID_{i-1}, s_{i-1}, m_{i-1})$ and (ID_i, s_i, r_i, m_i) from the signer I_i .
2. For $j = i, i-1, \dots, 3, 2$; compute $R'_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod{p}$, $T_j = R'_j \cdot r_j^{-1} \pmod{q}$, and $r_{j-1} = T_j \cdot (h_1(m_j || ID_j))^{-1} \pmod{q}$ by using I_j 's public keys y_j . Let $j = j-1$ and repeat step 2.
3. Finally compute $R'_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod{p}$, and $T_1 = R'_1 \cdot r_1^{-1} \pmod{q}$, and verify $T_1 \stackrel{?}{=} h_1(m_1 || ID_1) \pmod{q}$.

Our multisignature based on ElGamal-type signature has a feature that each signer has only one pair of a public key and a secret key.

4.2 RSA Based Scheme

Here we present our multisignature scheme based on RSA multisignature([4]).

Initialization: An authenticated center publishes small primes $\{r_l\} = \{2, 3, 5, \dots\}$. A signer I_i with identity information ID_i generates two large primes p_i and q_i secretly, and computes public keys $n_{i,l}$ and $e_{i,l} \in \mathbb{Z}_{n_{i,l}}^*$ in such a way that $n_{i,l} = p_i q_i r_l$, $L_{i,l} = LCM((p_i - 1), (q_i - 1), (r_l - 1))$, $e_{i,l} d_{i,l} = 1 \pmod{L_{i,l}}$, by using $\{r_l\}$. Signer I_i publishes all his public keys $n_{i,l}$, $e_{i,l}$ and r_l like Table 4.2.

In RSA-based multisignature, both operations in $\mathbb{Z}_{n_{i,l}}$ $\otimes$ and $\odot$ are set to $\oplus$ (EOR), and I_i 's signature sgn_i is just the next input to I_{i+1} 's signature generation: sgn_i is not divided into two parts.

Signature generation:

1. The first signer I_1 generates a signature on an original message m_1 : select a minimum number n_{1,l_1} such that $n_{1,l_1} > h_1(m_1 || ID_1)$ and compute $sgn_1 = (h_1(m_1 || ID_1))^{d_{1,l_1}} \pmod{n_{1,l_1}}$. Then send (ID_1, m_1, l_1, sgn_1) as a signature on m_1 to the next.
2. A signer I_j receives $m_1, m_2, \dots, m_{j-1}$ from I_{j-1} . If $j > 2$, patch the message $M_{1,2,\dots,j-1} = Patch(m_1, m_2, \dots, m_{j-1})$, modify it to $M_{1,2,\dots,j}$. Then I_j

- generates a signature on $m_j = Diff(M_{1,2,...,j-1}, M_{1,2,...,j-1,j})$: select a minimum number n_{j,l_j} such that $n_{j,l_j} > sgn_{j-1} \oplus h_1(m_j || ID_j)$, and compute $T = sgn_{j-1} \oplus h_1(m_j || ID_j)$, and $sgn_j = T^{d_{j,l_j}} \pmod{n_{j,l_j}}$.
3. A multisignature of $M_{1,2,...,i} = Patch(m_1, m_2, ..., m_i)$ by $I_1, \dots, I_{i-1}$ and I_i is given by (ID_1, l_1, m_1) , (ID_2, l_2, m_2) , $\dots$, and (ID_i, l_i, m_i, sgn_i) .

Table 1. I_i 's pairs of secret key and public key

l	1	2	$\dots \dots$
r_l	r_1	r_2	$\dots \dots$
public keys	$(n_{i,1}, e_{i,1})$	$(n_{i,2}, e_{i,2})$	$\dots \dots$
secret keys	$d_{i,1}$	$d_{i,2}$	$\dots \dots$

Signature verification:

1. The verifier receives (ID_1, l_1, m_1) , (ID_2, l_2, m_2) , $\dots$, (ID_i, l_i, m_i, sgn_i) from a signer I_i .
2. For $j = i, i-1, \dots, 2$; compute $T' = (sgn_j)^{e_{j,l_j}} \pmod{n_{j,l_j}}$, and $sgn_{j-1} = h_1(m_j || ID_j) \oplus T'$ by using I_j 's public key (n_{j,l_j}, e_{j,l_j}) . Let $j = j-1$ and repeat step2.
3. Compute $T' = sgn_1^{e_{1,l_1}} \pmod{n_{1,l_1}}$ by using I_1 's public key (n_{1,l_1}, e_{1,l_1}) , and check $T' \stackrel{?}{=} h_1(m_1 || ID_1)$.

Our multisignature based on RSA has the following features: 1. The size of multisignature keeps low even if the number of signers increases, compared with DLP based scheme. 2. It is necessary for each signer to have plural pairs of secret and public key.

4.3 Performance Evaluation

We evaluate our two multisignature schemes from a point of view of computation amount, the signature size and the number of rounds, where the signature size means that the final multisignature by $I_1, \dots, I_i$, and the number of rounds means how many times the process to generate the signature runs among all signers. There has not been proposed a multisignature with message flexibility, order flexibility and order verifiability. One primitive scheme with message flexibility is a simple chain of signature: each signer makes a signature on his own modification and sends it together with the previous signer's signature. Apparently it does not satisfy order verifiability. We also compare our schemes with the primitive scheme. For a simple discussion, we assume the following conditions: 1. a primitive arithmetic of binary methods([5]) is used for computation of exponentiation; 2. we denote the number of signers and the computation time for one n -bit modular multiplication by i and $M(n)$, respectively, where $M(n) = (\frac{m}{n})^2 M(m)$; 3. two primes p and q are set to 1024 and 160 bits respectively, in DLP-based signature schemes; 4. two primes p_j and q_j are set to 512 bits, and r_l is less than 10 bits in RSA-based signature schemes.

Table 2. Performance of DLP-based multisignature schemes

	Computation amount $\#M(1024)$		Signature size (bits)	#rounds	Features
	I_i 's signature generation	signature verification			
Our scheme	243	$483i$	$160(i+1)$	1	MF, OF, OV
Primitive scheme	242	$483i$	$320i$	1	MF
Scheme([11])	242	$481 + 241i$	$160 + 1024i$	1	—
Scheme([1])	483	1778	2048	2	OV

MF: Message Flexibility, OF: Order Flexibility, OV: Order Verifiability

Table 3. Performance of RSA based signatures

	Computation amount $\#M(1024)$		Signature size (bits)	#rounds	Features
	I_i 's signature generation	signature verification			
Our scheme	1536	$9i$	$1024 + 10i$	1	MF, OF, OV
Primitive scheme	1536	$9i$	$1024i$	1	MF

DLP based-multisignature schemes are mainly classified into two types, one-round scheme([11]) and two-round scheme in Section 2. Generally, the signature verification phase in two-round scheme is more simple than one-round scheme. However the signature generation phase in two-round scheme, which runs twice through all signers, is rather complicated. Here we compare our scheme with the primitive scheme, one-round scheme([11]) and two-round scheme([1]) Table 2 shows performance of 4 schemes. From Table 2, we see that only the computation amount for signature verification increases, and the signature size is even reduced, compared with the same one-round multisignature. Therefore our protocol can realize three features with message flexibility, order flexibility, and order verifiability only with negligible additional computation amount in signature generation.

Here we compare our RSA-based multisignature scheme with the primitive scheme. Table 2 shows performance of two schemes. From Table 2, we see that our protocol can realize three features, message flexibility, order flexibility, and order verifiability, with neither additional computation amount nor signature size.

5 Security Consideration

In this section, we discuss the security relation between our DLP based multisignature scheme and DLP. We assume that all signers except for an honest signer I_n collude in attacks: attackers use all secret keys $x_j (j \neq n)$, random numbers k_j , public information like public keys, all messages $m_1, \dots, m_n \in \mathbb{Z}$ and valid partial signatures. By using these informations, attackers try to forge I_i 's signatures. For simplicity, we denote the sequence $x_1, x_2, \dots, x_n$ by $x_{[1,n]}$ and the

sequence $x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ by $x_{[1,n,i]}$, where $1 \leq i \leq n$. We also denote $x_1, x_2, \dots, x_n \in \mathbb{Z}_q$ by $x_{[1,n]} \in \mathbb{Z}_q$. In our security proof, we use the polynomial-time truth-table ($\leq_{k-tt}^{fp}$) reducibility of the function version ([10]), which discusses passive attacks. In $\leq_{k-tt}^{fp}$ only k non-adaptive queries to an oracle are allowed.

5.1 Functions

First we define some functions.

Definition 1. $\text{DLP}(X, g, p, q)$ is the function that on input two primes p, q with $q|(p-1)$, $X, g \in \mathbb{Z}_p^*$ outputs $a \in \mathbb{Z}_q$ such that $X = g^a \pmod{p}$ if such $a \in \mathbb{Z}_q$ exists.

We define the function **Forge** that forges I_n 's valid signature (r_n, s_n) on $m_{[1,n]}$ in order $I_{[1,n]}$ by using available public information, a signature on $m_{[1,n-1]}$ by $I_{[1,n-1]}$ and available secret data like $x_{[1,n-1]}$ and $k_{[1,n-1]}$ for attackers $I_{[1,n-1]}$.

Definition 2. $\text{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n)$ is the function that on input two primes p, q with $q|(p-1)$, $y_n, g \in \mathbb{Z}_p^*$, $s_{[1,n-1]}, r_{n-1}, x_{[1,n-1]}, k_n \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_n \in \mathbb{Z}$, outputs $(r_n, s_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ such that $t_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod{p}$, $T_j = t_j \cdot r_j^{-1} \pmod{q}$, and $r_{j-1} = T_j \cdot (h_1(m_j || ID_j))^{-1} \pmod{q}$ for $j = n, n-1, \dots, 3, 2$ and that $t_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod{p}$ and $T_1 = t_1 \cdot r_1^{-1} \pmod{q}$ if such $(r_n, s_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ exists.

Next we define the function **Exclude** that forges I_n 's valid signature (s'_n, k_n) on $m_{[1,n,n-1]}$ in order $I_{[1,n,n-1]}$ by using available public information, a signature on $m_{[1,n]}$ by $I_{[1,n]}$ and available secret data $x_{[1,n-1]}$ and $k_{[1,n-1]}$ for attackers $I_{[1,n-1]}$.

Definition 3. $\text{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n)$ is the function that on input two primes p, q with $q|p-1$, $g, y_n \in \mathbb{Z}_p^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]}, r_n, s_{[1,n]} \in \mathbb{Z}_q^*$, output $(s'_n, k_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ such that $R_n = g^{k_n} \pmod{p}$, $r'_n = (h_1(m_n || ID_n) \times r_{n-2})^{-1} R_n \pmod{q}$, and $s'_n = (x_n r'_n + 1) k_n^{-1} \pmod{q}$, for $j = n-2, \dots, 2$: $t_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod{p}$, $T_j = t_j \cdot r_j^{-1} \pmod{q}$, and $r_{j-1} = T_j \cdot (h_1(m_j || ID_j))^{-1} \pmod{q}$, and that $t_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod{p}$, $T_1 = t_1 \cdot r_1^{-1} \pmod{q}$ if such $(s'_n, k_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ exists.

Next we define the function **SWAP** that forges valid multisignature on $m_{[1,n-2]}$, m_n, m_{n-1} in order $I_{[1,n-2]}, I_n, I_{n-1}$ by using available public information, a valid multisignature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$ and available secret data $x_{[1,n-1]}$ and $k_{[1,n-1]}$ for attackers $I_{[1,n-1]}$. From the assumption that $I_{[1,n-1]}$ are attackers, the function **SWAP** that forges I_n 's signature (r_n, s_n) on $m_{[1,n-2]}$, m_n, m_{n-1} in order $I_{[1,n-2]}, I_n, I_{n-1}$ for a valid signature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$ is just the same as the function that computes **Exclude** and adds attacker I_{n-1} 's signature on $m_{[1,n-2]}$, m_n, m_{n-1} in order $I_{[1,n-2]}, I_n, I_{n-1}$. Oppositely, the function **Exclude** is just the same as the function that for a valid signature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$, computes **SWAP** and outputs only I_n 's multisignature (r_n, s_n) . Therefore the following theorem holds.

Theorem 1. $\text{SWAP} \equiv_{1\text{-}tt}^{fp} \text{Exclude}$.

For the sake of the following proof, we define the function **SIGN** that generates a valid signature $(r_n, s_{[1,n]})$ on messages $m_{[1,n]}$ by signers $I_{[1,n]}$ by using all secret data $x_{[1,n]}$ and $k_{[1,n]}$ of signers $I_{[1,n]}$. This function means just the signature generation function. Apparently it is easy to compute **SIGN**.

Definition 4. $\text{SIGN}(g, p, q, x_{[1,n]}, k_{[1,n]}, m_{[1,n]}, ID_{[1,n]})$ is the function that on input two primes p, q with $q|(p-1)$, $g \in \mathbb{Z}_p^*$, $x_{[1,n]}, k_{[1,n]} \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, output $r_n, s_{[1,n]} \in \mathbb{Z}_q^*$ such that for $j = n, \dots, 3, 2$, $t_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod{p}$, $T_j = t_j \cdot r_j^{-1} \pmod{q}$ and $r_{j-1} = T_j \cdot (h_1(m_j || ID_j))^{-1} \pmod{q}$ and that $t_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod{p}$, $T_1 = t_1 \cdot r_1^{-1} \pmod{q}$ if such $r_n, s_{[1,n]} \in \mathbb{Z}_q$ exists.

5.2 Reduction among Functions

Here we show our results. First we set functions ψ_i to give the i -th element, $\psi_i(a_{[1,n]}) = a_i (i \leq n)$.

Theorem 2. $\text{Forge} \equiv_{1\text{-}tt}^{fp} \text{DLP}$

proof: First we show that $\text{Forge} \leq_{1\text{-}tt}^{fp} \text{DLP}$. For inputs $(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1})$ of **Forge**, fix $k_n \in \mathbb{Z}_q$ and set $R_n = g^{k_n} \pmod{p}$, $r_n = r_{n-1}^{-1} \cdot h_1(m_n || ID_n)^{-1} \cdot R_n \pmod{p}$. Then

$$\begin{aligned} \text{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n) \\ = (r_n, (\text{DLP}(y_n, g, p, q) r_n + 1) k_n^{-1} \pmod{q}). \\ = (r_n, s_n). \blacksquare \end{aligned}$$

Next we show that $\text{DLP} \leq_{1\text{-}tt}^{fp} \text{Forge}$. For input (y_n, g, p, q) of **DLP**, fix $k_{[1,n]} \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]} \in \mathbb{Z}_q^*$, and set

$$(r_{n-1}, s_{[1,n-1]}) = \text{SIGN}(g, p, q, x_{[1,n-1]}, k_{[1,n-1]}, m_{[1,n-1]}, ID_{[1,n-1]}),$$

which is computed in time polynomial from the definition. Then

$$\text{DLP}(y_n, g, p, q)$$

$$= (\psi_2(\text{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n)) \cdot k_n - 1) r_n^{-1},$$

where $r_n = \psi_1(\text{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n))$ and $y_n = g^{x_n}$. $\blacksquare$

Therefore we get $\text{DLP} \equiv_{1\text{-}tt}^{fp} \text{Forge}$. $\blacksquare$

Theorem 3. $\text{Exclude} \equiv_{1\text{-}tt}^{fp} \text{DLP}$

proof: First we show that $\text{Exclude} \leq_{1\text{-}tt}^{fp} \text{DLP}$. For inputs $(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n)$ of **Exclude**, fix $k_n \in \mathbb{Z}_q$, and set $R_n = g^{k_n} \pmod{p}$, and $r'_n = r_{n-2}^{-1} \cdot h_1(m_n || ID_n)^{-1} \cdot R_n \pmod{p}$. Then

$$\begin{aligned} \text{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n, k_n) \\ = ((\text{DLP}(y_n, g, p, q) r'_n + 1) k_n^{-1} \pmod{q}, k_n) \blacksquare \end{aligned}$$

Next we show that $\text{DLP} \leq_{1-tt}^{fp} \text{Exclude}$. For inputs (y_n, g, p, q) of DLP, fix $k_{[1,n-1]} \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]} \in \mathbb{Z}_q^*$, and set

$$(r_{n-2}, s_{[1,n-2]}) = \text{SIGN}(g, p, q, x_{[1,n-2]}, k_{[1,n-2]}, m_{[1,n-2]}, ID_{[1,n-2]}),$$

which is computed in time polynomial from the definition. Then

$$\text{DLP}(y_n, g, p, q) = (s'_n \cdot k_n - 1) \cdot r'^{-1}_n, \text{ where}$$

$$s'_n = \psi_1(\text{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n)),$$

$$k_n = \psi_2(\text{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n)),$$

$$R_n = g^{k_n} \pmod{p}, \text{ and } r'_n = (r_{n-2} \cdot h_1(m_n || ID_n))^{-1} \cdot R_n \pmod{q}. \blacksquare$$

Then we get $\text{DLP} \equiv_{1-tt}^{fp} \text{Exclude}$. ■

6 Further Discussion

We discuss how to add the following feature to our multisignature scheme.

Robustness: If the signature verification fails, then prevent such an unauthentic message from damaging a receiver.

We realize robustness by combining our multisignature with an encryption function. So we call it *multisigncrypt*. Multisigncrypt nd has a feature that a message cannot be recovered if the signature verification fails, in addition to message flexibility, order flexibility, and order verifiability. Therefore a multisigncrypt can prevent computer virus mixed into a message from damaging a receiver since unauthentic message can not be recovered.

6.1 Multisigncrypt Scheme

For simplicity, we present the multisigncrypt scheme by using our basic multisignature scheme.

Initialization: A center publishes two hash functions h_1 and h_2 , and an encryption and the decryption function, $E(K_i, m_i)$ and $D(K_i, C_i)$, in addition to initialization in basic multisignature scheme, where h_2 is used for computing a session key K_i for E and D , and C_i is a cipher text.

Signature generation:

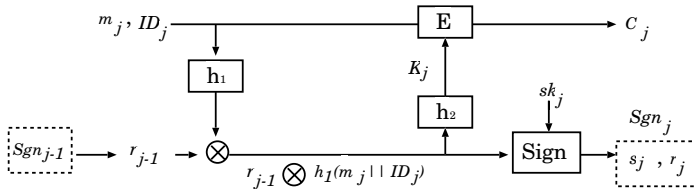


Fig. 3. I_j 's signature generation

1. The first signer I_1 computes

$$sgn_1 = \text{sign}(sk_1, h_1(m_1 || ID_1)) = (r_1, s_1),$$

where sgn_1 is divided into two parts of r_1 and s_1 in the same way as Section 3, generates a session key K_1 ,

$$K_1 = h_2(h_1(m_1 || ID_1)),$$

and encrypts $m_1 || ID_1$ by an encryption function E ,

$$C_1 = E(K_1, m_1 || ID_1),$$

and sends (ID_1, s_1, r_1, C_1) to the next signer I_2 .

2. A signer I_j verifies the signature from I_{j-1} , $m_1, \dots, m_{j-1}$ according to the verification step in the next page, and modifies $M_{1, \dots, j-1} = \text{Patch}(m_1, \dots, m_{j-1})$ to $M_{1, \dots, j}$. Then I_j generates a signature on the difference $m_j = \text{Diff}(M_{1, \dots, j-1}, M_{1, \dots, j-1, j})$: compute

$$sgn_j = \text{Sign}(sk_j, r_{j-1} \otimes h_1(m_j || ID_j)) = (r_j, s_j),$$

$$K_j = h_2(r_{j-1} \otimes h_1(m_j || ID_j)),$$

and encrypts $m_j || ID_j$ by using the session key K_j ,

$$C_j = E(K_j, m_j || ID_j).$$

3. A multisignature on $M_{1,2,\dots,i} = \text{Patch}(m_1, m_2, \dots, m_i)$ by $I_1, \dots, I_i$ is given by $(ID_1, s_1, C_1), (ID_2, s_2, C_2), \dots, (ID_i, s_i, r_i, C_i)$.

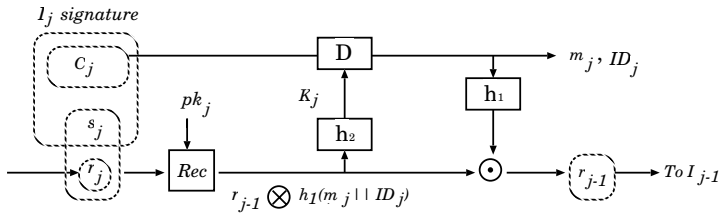


Fig. 4. I_j 's signature verification step

Signature verification:

1. The verifier receives $(ID_1, s_1, C_1), \dots, (ID_{i-1}, s_{i-1}, r_{i-1}, C_{i-1}), (ID_i, s_i, r_i, C_i)$ from the signer I_i .

2. For $j = i, \dots, 3, 2$: compute

$$T_j = \text{Rec}(pk_j, (s_j, r_j)), \text{ and } K_j = h_2(T_j),$$

and decrypts m_j and ID_j by

$$m'_j || ID'_j = D(K_j, C_j).$$

If $ID'_j \stackrel{?}{=} ID_j$ holds, then accept the signature and recover r_{j-1} ,

$$r_{j-1} = T_j \odot h_1(m'_j || ID'_j).$$

Set $j = j - 1$ and repeat step 2.

3. Compute

$$T_1 = \text{Rec}(pk_1, (s_1, r_1)) \text{ and } K_1 = h_2(T_1),$$

and decrypt m_1 and ID_1 by

$$m'_1 || ID'_1 = D(K_1, C_1).$$

If $h_1(m'_1 || ID'_1) \stackrel{?}{=} T_1$ holds, then accept the signature and finally patch all messages,

$$M_{1, \dots, i} = \text{Patch}(m_1, \dots, m_i).$$

In both cases of DLP- and RSA-based multisignature schemes, we can also add the feature of Robustness in the same way as the above.

7 Conclusion

In this paper, we have proposed a new multisignature scheme suitable for circulating messages through Internet. Our multisignature scheme realizes the three features, Message flexibility, Order flexibility and Order verifiability, maintaining both signature size and computation amount in signature generation/verification low: only the computation amount for the signature verification increases, and the signature size is even reduced compared with one round previous multisignature scheme. We have also proposed the multisigncrypt scheme, which realizes Robustness in addition to Message flexibility, Order flexibility and Order verifiability. Furthermore, we have proved the following equivalences between our DLP-based multisignature and DLP in some typical attacks by using the reducibility of functions.

1. $\text{FORGE} \equiv_{tt}^{fp} \text{DLP}$
2. $\text{SWAP} \equiv_{tt}^{fp} \text{DLP}$
3. $\text{EXCLUDE} \equiv_{tt}^{fp} \text{DLP}$

References

1. M. Burmester, Yvo Desmedt, Hiroshi Doi, Masahiro Mambo, Eiji Okamoto, Mitsuru Tada, and Y. Yoshifuji, "A Structured ElGamal-Type Multisignature Scheme", *Advances in Cryptology-Proceedings of PKC'2000*, Lecture Notes in Computer Science, (2000), Springer-Verlag, 466-482.
2. "Specification for a digital signature standard", National Institute for Standards and Technology, Federal Information Standard Publication XX, draft (1991).
3. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Trans. Inform. Theory*, Vol. IT-31 (1985), 469-472.
4. K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures". *NEC J.Res.Dev.*71(Oct.1983).
5. D. E. Knuth, *The art of computer programming, vol. 2, Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, Mass. 1981.
6. A. Miyaji, "Another countermeasure to forgeries over message recovery signature", *IEICE Trans.*, Fundamentals. vol. E80-A, No.11(1997), 2192-2200.
7. K. Nyberg and R. A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem", *Designs Codes and Cryptography*, 7(1996), 61-81.
8. T. Okamoto, "A digital Multisignature Scheme Using Bijective Public-key Cryptosystems", *ACM Trans. on Computer Systems*, Vol.6, No.8(1988), 432-441.
9. R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol.21, No.2(1978), 120-126.
10. K. Sakurai and H. Shizuya "Relationships among the computational powers of breaking Discrete Log cryptosystem", *Advances in Cryptology-Proceedings of Eurocrypt'95*, Lecture Notes in Computer Science, 921(1995), Springer-Verlag, 341-355. (J. Cryptology,11 (1998), 29-43.)
11. A. Shimbo, "Multisignature Schemes Based on the Elgamal Scheme", *The 1994 Symposium on Cryptography and Information Security*, **SCIS94-2C**, Jan. 1994.
12. C. P. Schnorr, "Efficient signature generation by smart cards", *Journal of cryptology*, 4(1991), 161-174.
13. T. Saito, "A Multiple Signature Scheme Enabling a Specified Signer's Order", *The 1997 Symposium on Cryptography and Information Security*, **SCIS97-33A**, Jan. 1994.

Light Weight Broadcast Exclusion Using Secret Sharing

Natsume Matsuzaki¹, Jun Anzai¹, and Tsutomu Matsumoto²

¹ Advanced Mobile Telecommunications Security Technology
Research Laboratories Co., Ltd.

BENEX S3 Building 12F
3-20-8, Shinyokohama, Kohoku-ku, Yokohama, 222-0033 Japan
{matuzaki, anzai}@ams1.co.jp

² Division of Artificial Environment and Systems, Yokohama National University
79-5, Tokiwadai, Hodogaya, Yokohama, 240-8501 Japan
tsutomu@mlab.jks.ynu.ac.jp

Abstract. In this paper, we examine a *broadcast exclusion problem*, i.e., how to distribute an encryption key over a channel shared by n entities so that all but k excluded entities can get the key. Recently, J. Anzai, N. Matsuzaki and T. Matsumoto proposed a scheme that provides a solution to the broadcast exclusion problem. Their solution is to apply $(k + 1, n + k)$ threshold cryptosystems. In this scheme, the transmission overhead is $O(k)$ and each entity holds a fixed amount of secret key. However, each entity must compute the encryption key with $k + 1$ modular exponentiations. Therefore, a device with low computing power (e.g., a mobile terminal or a smart card) cannot calculate the broadcast key within a reasonable time. In this paper, we propose a new scheme in which each entity computes the key with only two modular exponentiations, regardless of n and k . We accomplish this by assuming a trusted key distributor, while retaining the advantages of Anzai-Matsuzaki-Matsumoto scheme, i.e., the transmission overhead is $O(k)$, and each entity holds a fixed amount of secret key regardless of n and k .

1 Introduction

Background. A *broadcast encryption* allows a center to send the same message simultaneously and privately over a broadcast channel to all authorized entities. Cable and satellite Pay-TV, Internet multicasts and group telecommunications (e.g., private mobile radio or taxi radio) are typical examples of systems that can use the broadcast encryption. In these systems, it is desired a secure and fast method to distribute a shared key (which is called *broadcast key* in this paper) with all the authorized entities.

This paper focuses on a *broadcast exclusion problem* (also known as a *black-listing problem*) that is a kind of the broadcast encryption. The broadcast exclusion problem is how to transmit a broadcast key over a broadcast channel shared by n entities so that all but k excluded entities can get the broadcast key. Without the exclusion, an unauthorized entity could eavesdrop the secret broadcasting

using the telecommunications terminal which an authorized entity has lost. It is important to exclude this terminal as quickly as possible. Also, the broadcast exclusion will prevent entities from using paid service like Pay-TV and Internet access without paying a charge.

Previous works. One simple method of the broadcast exclusion is that a center sends a new broadcast key to each entity except the excluded entities, as encrypted form by a secret key of each non-excluded entity. This method requires a small amount of storage for each entity, since each entity stores only one secret key. However, the center must transmit $n - k$ keys. This means that a large amount of transmission is required when n is large, where n is the number of all entities on the broadcast channel.

Recently, two major works on the broadcast exclusion problem have been presented. One method proposed by Kumar, Rajagopalan and Sahai [8] uses error-correcting codes without computational assumptions.

Another method proposed by Anzai, Matsuzaki and Matsumoto [1] uses $(k + 1, n + k)$ threshold cryptosystems in order to simultaneously exclude up to k entities. The transmission overhead is $O(k)$ regardless of n , and each entity has only one key. Moreover, there is no requirement for a fixed-privileged key distributor. However, this scheme requires each entity to compute the broadcast key with $k + 1$ modular exponentiations. The parameter k should be set a large integer to strengthen against a conspiracy attack. Using a large value for k , devices with low computing power (e.g., mobile terminals, smart cards, etc.) take a long time to calculate the broadcast key.

Our result. This paper proposes a new scheme in which each entity computes the broadcast key with two modular exponentiations, regardless of n and k . This allows even low-powered devices to calculate the broadcast key in a short period of time. Our scheme uses $(k + 1, n + k)$ threshold cryptosystems, similar to Anzai-Matsuzaki-Matsumoto scheme. The transmission overhead is $O(k)$ and each entity stores a fixed amount of secret key regardless of n and k . The only additional assumption is the introduction of a fixed-privileged key distributor, that is a reasonable assumption in actual applications.

Organization. Section 2 describes previous works in this area and states our goal. Section 3 describes our target system and some assumptions. Section 4 shows our approach and the basic scheme, and analyzes the problems. Section 5 proposes our scheme as the result of considerations in section 4. Section 6 discusses the security of our proposed scheme. Section 7 describes the other considerations. Section 8 evaluates the performance and features of our scheme.

2 Previous Works and Our Goal

Related works. In [7], Fiat and Naor introduced a general idea of a “*broadcast encryption*” wherein a center securely broadcasts a *broadcast key* or a message to selected subsets of entities. Also, they proposed a scheme which is resilient

to any coalition of k entities, wherein every entity stores $O(k \log k \log n)$ keys and the center broadcasts $O(k^2(\log k)^2 \log n)$ messages. A number of works have followed [2][3][5][9], which focused mainly on the trade-off between the number of broadcasts and the number of keys held by each entity.

On the other hand, many researchers have investigated a *broadcast exclusion problem* that is a kind of the broadcast encryption. Two research groups of Wallner, Harder and Agee [12] and Wong, Gouda and Lam [13] proposed a hierarchical key distribution scheme using a balanced binary tree. In this scheme, the transmission overhead is $O((degree - 1) \times \log n)$ and the number of keys for each entity is $O(\log n)$, where n is the number of entities on the broadcast channel and *degree* is the number of entities in the bottom subgroup of hierarchical tree. Canetti, Garay, Itkis, Micciancio, Naor and Pinkas proposed an extended method [4] which reduces the amount of transmission. And, Canetti, Malkin and Nissim [5] studied the trade-off between the number of broadcasts and the number of keys held by each entity.

Recently, two works on the broadcast exclusion have been presented: one by Kumar, Rajagopalan and Sahai [8] and one by Anzai, Matsuzaki and Matsumoto [1]. In Kumar-Rajagopalan-Sahai scheme, each entity has an individual subset of keys. Redundant pieces of a message using an error-correcting code, are encrypted by keys belonging to non-excluded entities. The transmission overhead is $O(k^2)$ regardless of n . And, the number of keys stored by each entity is $O(k \times \log n)$, still depending on the number of entities in the group.

Anzai-Matsuzaki-Matsumoto scheme. Anzai, Matsuzaki and Matsumoto proposed a scheme [1] using $(k + 1, n + k)$ threshold cryptosystems to exclude up to k entities simultaneously. Their basic scheme is as follows:

- A center divides a secret key S into $n + k$ shadows with threshold $k + 1$ using a well-known secret sharing scheme [11]. And, he distributes n shadows to each of n entities in a secure manner. He stores the remaining k shadows as spare shadows. $(k + 1, n + k)$ secret sharing scheme is a probabilistic mapping of a secret to $n + k$ shadows, such that
 - (a) the secret can be reconstructed from any $k + 1$ out of $n + k$ shadows, and
 - (b) no subset of k shadows reveals any partial information about the secret.
- When the center excludes d entities, he broadcasts k shadows, i.e., d shadows of excluded entities plus $k - d$ spare shadows. The transmission overhead is $O(k)$ regardless of n .
- A non-excluded entity can recover S using $k + 1$ shadows, i.e., his secret key plus the broadcast shadows. The excluded entity cannot recover S because one of the broadcast shadows is his own secret key, and he can get only k shadows.

In order to exclude other entities in the next round, a center shall generate a new secret key S' and distributes new shadows of S' to each of the entities secretly, since the secret key S has come out to all the non-excluded entities, who include the excluded entities in the next round. The center cannot do frequent key renewal because the distribution of shadows to each of entities takes a long time

when n is large. They modify their basic scheme to broadcast the shadows $\{s_j\}$ lifting to the exponentiation part with a random number, i.e., $\{g^{r \times s_j} \bmod p\}$. And, each non-excluded entity obtains the broadcast key $g^{r \times S} \bmod p$, gathering $k + 1$ shadows of the secret S on the exponentiation part. This approach is similar to the *threshold cryptosystems* proposed by Desmedt and Frankel in [6]. The transmission overhead is $O(k)$ regardless of n . And, each entity needs to hold only one secret. Therefore, the center can do the quick broadcast exclusion when n is large. And, their scheme is suitable for an entity with a small memory, compared with the other schemes. Moreover, any entity can exclude any other entity, using $\{g^{s_j} \bmod p\}$ on a public bulletin board.

However, this scheme requires each entity to compute the broadcast key with $k + 1$ modular exponentiations. The parameter k should be set a large integer to strengthen against a conspiracy attack. Using a large value for k , it takes a long time for entities with low computing power (such as a mobile terminal and a smart card) to calculate the broadcast key.

Our goal and approach. Our goal is to propose a scheme in which the number of modular exponentiations for each entity can be kept small (regardless of n and k). Our approach is to distribute the shadows of excluded entities on the multiplicative group of the base field, with unknown parameters to conceal the shadows. To achieve our goal, we compromise to introduce a fixed-privileged key distributor who knows all secrets and determines the excluded entities. We consider the assumption of the key distributor is reasonable in several actual applications.

3 Target Model and Assumptions

Our target model consists of the following components:

Key distributor: A trusted party who decides system parameters. He sets and stores all of the entities' secret keys. He alone can decide who should be excluded, and distributes the broadcast key for all the valid entities, except for the excluded entities.

Entity i : A user or a terminal sharing the same broadcast channel. Entity i stores a secret key s_i . We assume that the number of total entities is n . Let $\Phi = \{1, 2, \dots, n\}$ be the set of the entities.

Excluded entity j : An entity to be excluded by the key distributor. Let $\Lambda(\subset \Phi)$ be the set of excluded entities, having d entities.

Valid entity v : An entity who is not an excluded entity.

In this target model, we make the following system assumptions:

1. All entities trust the key distributor. The key distributor doesn't do anything illegal.
2. All entities have simultaneous access to the data that the key distributor broadcasts.

3. The broadcast channel is insecure, i.e., anyone can see the data on the channel.
4. The number of all entities is very large compared with the number of excluded entities, i.e., d is a small percentage of n .

We also make the following assumptions on the security:

1. A discrete logarithm problem is computationally hard to solve.
2. In $(k + 1, n + k)$ threshold cryptosystems, anyone with less than or equal to k shadows cannot get any information about the secret.
3. Excluded entities may conspire to get the broadcast key.
4. Excluded entities may publish their secret information to damage the system security.
5. Valid entities do not conspire with excluded entities. If this assumption is not satisfied, the excluded entity can get the broadcast key from the valid entity.

4 Our Approach

We modify Anzai-Matsuzaki-Matsumoto scheme as to distribute shadows of excluded entities on the multiplicative group of the base field, to reduce the number of modular exponentiations. In this section, we show the basic scheme and its problems.

4.1 Basic Scheme

The scheme contains three phases: the system setup, the key distribution by the key distributor and the key calculation on each valid entity.

[System Setup]

First, a key distributor does system setup. He generates system parameters and secret keys of all entities as follows:

1. The key distributor decides the following system parameters and publishes them:
 - n : the number of entities
 - d : the number of excluded entities
 - k : an integer such that $0 \leq d \leq k < n$
 - p : a large prime number such that $n + k < p$
 - q : a large prime number such that $q \mid p - 1$
 - g : a q^{th} root of unity over $\mathbf{Z}_p$
2. The key distributor generates a system secret $S \in \mathbf{Z}_q$ and stores it secretly. The key distributor divides S into $n + k$ shadows with threshold $k + 1$, using the well-known secret sharing scheme by Shamir [11]. Let s_i ($i = 1, \dots, n + k$) be the $n + k$ shadows.

3. The key distributor distributes s_i to the corresponding entity i in a secure manner.

[Key Distributor]

Whenever the key distributor wants to re-new the broadcast key excluding selected entities, he generates and broadcasts the data as follows:

1. The key distributor selects a random number $r \in \mathbf{Z}_q$, and calculates *preparation data*:

$$X = g^r \bmod p. \quad (1)$$

2. The key distributor decides entities to be excluded. Let Λ be the set of excluded entities, and d be the number of excluded entities.
3. The key distributor picks $k - d$ integers arbitrary from a set $\{n + 1, \dots, n + k\}$. Let Θ be the set of chosen integers. Then, the key distributor calculates *k exclusion data*:

$$m_j = r \times s_j \bmod q. \quad (j \in \Lambda \cup \Theta) \quad (2)$$

4. The key distributor broadcasts the following *broadcast data* to all entities:

$$b = X \parallel \{(j, m_j) \mid j \in \Lambda \cup \Theta\}, \quad (3)$$

where $\parallel$ indicates the concatenation of the data.

5. The key distributor calculates the broadcast key u as follows:

$$u = g^{r \times S} \bmod p. \quad (4)$$

[Valid Entity $v \notin \Lambda$]

Receiving the broadcast data from the key distributor, a valid entity v calculates the broadcast key u as follows:

$$u = X^{w1} \times g^{w2} \bmod p, \quad (5)$$

where

$$w1 = s_v \times L(v) \bmod q, \quad (6)$$

$$w2 = \sum_{j \in \Lambda \cup \Theta} (m_j \times L(j)) \bmod q, \quad (7)$$

and $L(j)$ is a Lagrange's interpolation polynomial:

$$L(j) = \prod_{t \in \Lambda \cup \Theta \cup \{v\} \setminus \{j\}} t / (t - j). \quad (8)$$

Every valid entity can obtain the common broadcast key u with only two modular exponentiations. The calculation amount of modular multiplication and modular addition in equation (5)(6)(7)(8) is negligible compared with a modular exponentiation. The broadcast key shown in equation (5) is the same as the key

of the key distributor as follows:

$$\begin{aligned}
 u &\equiv (g^r)^{s_v \times L(v)} \times g^{r \times \sum_{j \in A \cup \Theta} s_j \times L(j)} \pmod{p} \\
 &\equiv g^{r(s_v \times L(v) + \sum_{j \in A \cup \Theta} s_j \times L(j))} \pmod{p} \\
 &\equiv g^{r \times \sum_{j \in A \cup \Theta \cup \{v\}} s_j \times L(j)} \pmod{p} \\
 &\equiv g^{r \times S} \pmod{p}.
 \end{aligned}$$

[Excluded Entity $j \in A$]

On the other hand, an excluded entity j can not calculate the broadcast key u . The exclusion data includes the secret key s_j of excluded entity j . Using equation (5), he can gather only k shadows of the system secret S in the exponentiation part. Also, an outsider cannot get the broadcast key u because he can get only k broadcast shadows.

4.2 Discussion on Security of the Basic Scheme

In this section, we describe the basic scheme has several problems on the security.

Before the security analyses, we arrange the parameters as follows:

<i>System secret:</i>	S , all s_i and the random number r
<i>Entity secret:</i>	s_i
<i>Valid entity secret:</i>	broadcast key $u = g^{r \times S} \pmod{p}$
<i>Public data:</i>	public parameters n, k, p, q, g and all broadcast data including the preparation data $X = g^r \pmod{p}$, and the exclusion data m_j

First, we will discuss the security of the system secret S . Suppose that an entity knows the parameter S . He can obtain any broadcast key $u = X^S \pmod{p}$, using the public preparation data X , even if the key distributor does not want to share u with the entity.

Therefore, S must be concealed from all entities including valid entities. In basic scheme, however, S may be obtained by the following two attacks:

Secret publish attack: Suppose that the number of excluded entities is k , i.e., $d = k$. If all the excluded entities publish their secret shadows s_j , any valid entity can calculate S using $k + 1$ shadows: his own secret and k published shadows.

Spare shadow attack: Suppose that the key distributor uses the same spare exclusion data for each of R rounds. If the broadcast data b contains one exclusion data and $k - 1$ spare exclusion data for each of R rounds, an attacker gets $k \times R$ equations (shown in equation (2)) in $k - 1 + 2R$ unknowns: $k - 1$ spare shadows, R shadows of excluded entities and R random numbers. The attacker will get S using $k - 1 + R$ shadows, by solving as soon $k \times R$ exceeds $k - 1 + 2R$, for example $k = 4$ and $R = 2$.

To avoid the *secret publish attack*, d shall be less than k : i.e., $d < k$. And, to strengthen against the *spare shadow attack*, the key distributor shall generate new spare shadows of S for each round.

Next, we will discuss the security of the random number r . An excluded entity j can get the random number r from the exclusion data shown in equation (2), using his secret shadow s_j . So, we consider an attack as follows:

- r publish attack:** Suppose that an excluded entity publishes the random number r to all. Then, any entity can get k shadows from the exclusion data, using published r . So,
- any entity can get more than $k + 1$ shadows for at least two rounds, and
 - a valid entity can reconstruct the system secret S by adding his shadow to k shadows.

To conceal the random number r for any entity, we will introduce new unknown parameter t_j for each entity j into the exclusion data in equation (2), namely $m_j = r \times s_j + t_j \bmod q$. Nevertheless, if the key distributor sends the exclusion data of the same entities for more than two times, using the different random numbers, the random numbers can be solved with the conspiracy. For example, four unknown parameters (two random numbers, t_j and t_k) can be solved by four equations with the conspiracy of the excluded entity j and entity k . So, when the key distributor continues to exclude the entities who have been excluded in the last round, he should not re-send the exclusion data for them. The key distributor encrypts the broadcast data by the last broadcast key. The excluded entities in the last round who does not have the last broadcast key, are continuously excluded.

5 Proposed Scheme

As the result of considerations in the previous section, we propose the following scheme. Our proposed scheme can be based on an appropriate discrete logarithm problem defined over finite cyclic groups, including subgroups of Jacobians of elliptic curves and so on. In order to make the explanation be simple, we explain our scheme over a prime field $\mathbf{Z}_p$.

[System Setup]

1. The key distributor decides the following system parameters:
 - n : the number of entities
 - d : the number of excluded entities
 - k : an integer such that $0 \leq d < k < n$
 - p : a large prime number such that $n + k < q$
 - q : a large prime number such that $q \mid p - 1$
 - g : a q^{th} root of unity over $\mathbf{Z}_p$
 - $E(,)$: a symmetric cipher

2. The key distributor generates a system secret $S \in \mathbf{Z}_q$ and stores it secretly. The key distributor chooses a random polynomial F of degree $k + 1$ over $\mathbf{Z}_q$ subject to the condition $F(0) = S$, in the same way as Shamir's secret sharing scheme. Then, the key distributor computes n shadows s_i ($i = 1, 2, \dots, n$) as $F(i)$. Here, we assume $s_i \neq 0$ for every i . If $s_i = 0$, the key distributor changes the identity number from i to i' of the entity, wherein $s_{i'} \neq 0$.
3. The key distributor generates another system secret $T \in \mathbf{Z}_q$ and stores it secretly. The key distributor chooses another random polynomial G of degree $k + 1$ over $\mathbf{Z}_q$ subject to the condition $G(0) = T$. Then, the key distributor computes n shadows t_i ($i = 1, 2, \dots, n$) as $G(i)$.
4. The key distributor distributes $[s_i (\neq 0), f_i = g^{t_i/s_i} \bmod p]$ to the corresponding entity i in a secure manner.

[Key Distributor]

Let's assume that all valid entities have shared the broadcast key U' on round R . If $R = 0$, U' is a fixed universal secret. After that, the broadcast key on round $R + 1$ is distributed as follows.

1. The key distributor selects the random number $r \in \mathbf{Z}_q$ and calculates:

$$X = g^r \bmod p. \quad (9)$$

2. The key distributor decides which entities to exclude among the valid entities on round R . Let A be a set of d excluded entities.
3. The key distributor picks $k - d$ integers which are greater than $n + k \times (R - 1)$ and less than $n + k \times R$. Let Θ be the set of chosen integers. The key distributor calculates $s_l = F(l)$ and $t_l = G(l)$ for each $l \in \Theta$. Then, the key distributor calculates k exclusion data as follows:

$$M_j = r \times s_j + t_j \bmod q. \quad (j \in A \cup \Theta) \quad (10)$$

4. The key distributor calculates the broadcast data B and encrypts it by U' as follows:

$$E(U', B) = E(U', X \parallel \{(j, M_j) \mid j \in A \cup \Theta\}). \quad (11)$$

5. The key distributor calculates the broadcast key U :

$$U = g^{r \times S + T} \bmod p. \quad (12)$$

[Valid Entity $v \notin A$]

Receiving the encrypted broadcast data, valid entities on $R + 1$ round who are also valid entities on R round, decrypt it using U' . Then, a valid entity v calculates the broadcast key U for round $R + 1$ as follows:

$$U = (X \times f_v)^{W1} \times g^{W2} \bmod p, \quad (13)$$

where

$$W1 = s_v \times L(v) \bmod q, \quad (14)$$

$$W2 = \sum_{j \in A \cup \Theta} (M_j \times L(j)) \bmod q. \quad (15)$$

Every valid entity can obtain the common broadcast key U with only two modular exponentiations. The calculation amount of modular multiplication and modular addition in equation (13)(14) (15)(8) are negligible compared with a modular exponentiation. The broadcast key shown in equation (13) is the same as the key of the key distributor as follows:

$$\begin{aligned}
 U &\equiv (g^r \times g^{t_v/s_v})^{s_v \times L(v)} \times g^{r \times \sum_{j \in \Lambda \cup \Theta} s_j \times L(j) + \sum_{j \in \Lambda \cup \Theta} t_j \times L(j)} \pmod{p} \\
 &\equiv g^{r(s_v \times L(v) + \sum_{j \in \Lambda \cup \Theta} s_j \times L(j))} \times g^{t_v \times L(v) + \sum_{j \in \Lambda \cup \Theta} t_j \times L(j)} \pmod{p} \\
 &\equiv g^{r \times \sum_{j \in \Lambda \cup \Theta \cup \{v\}} s_j \times L(j)} \times g^{\sum_{j \in \Lambda \cup \Theta \cup \{v\}} t_j \times L(j)} \pmod{p} \\
 &\equiv g^{r \times S + T} \pmod{p}.
 \end{aligned}$$

Finally, we consider an attack to modify and forge the broadcast data. To avoid the attack, the key distributor puts his signature to the broadcast data. Here, we propose a scheme that requires three modular exponentiations to calculate the broadcast key with the signature verification, applying a message recovery signature technique proposed by Nyberg and Rueppel [10]. The key distributor calculates his message recovery signature using his secret key x_0 :

$$H = \text{hash}(j \parallel M_j), \quad j \in \Lambda \cup \Theta \quad (16)$$

$$Z = (H \times (-x_0) + r) \bmod q, \quad (17)$$

wherein hash is a public hash function. The key distributor generates the broadcast data B using Z instead of the preparation data X . Therefore, the transmission amount does not increase.

Receiving the broadcast data, the valid entity obtains the broadcast key U with three modular exponentiations, using the public key y_0 of the key distributor:

$$U = g^{z1} \times y_0^{z2} \times f_v^{z3} \bmod p, \quad (18)$$

where,

$$\begin{aligned}
 z1 &= Z \times s_v \times L(v) + \sum_{j \in \Lambda \cup \Theta} (M_j \times L(j)) \bmod q, \\
 z2 &= H' \times s_v \times L(v) \bmod q, \\
 z3 &= s_v \times L(v) \bmod q, \\
 H' &= \text{hash}(j \parallel M_j). \quad j \in \Lambda \cup \Theta
 \end{aligned}$$

The above equation is led from the equation:

$$\begin{aligned}
 X &\equiv g^r \pmod{p} \\
 &\equiv g^Z \times y_0^{H'} \pmod{p},
 \end{aligned}$$

that means that the preparation data X can be recovered from Z using the public key y_0 .

6 Security Analyses

First, we consider the security of the system secret S . Because the number of excluded entities is less than (not equal) the integer k , the *secret publish attack* is not possible. And, the key distributor uses fresh $k - d$ spare shadows s_j ($j \in \Theta$) for each round to strengthen against the *spare shadow attack*.

Concerning the security of the random number r , each excluded entity j (without the conspiracy) can not get r because new unknown parameter t_j is introduced in equation (10). Moreover, even if all excluded entities make the conspiracy, they cannot solve k simultaneous equations because the number of unknowns is $k + 1$: k secret shadows t_j and the random number r . Also, the excluded data of excluded entities are not re-used. So, any entity cannot solve simultaneous equations to get the random numbers. The key distributor continues to exclude the entities who have already been excluded last round, encrypting the broadcast data by the last broadcast key.

Next, we will discuss the security of $g^T \bmod p$. $g^T \bmod p$ and T must be concealed from all the entities. And, our proposed scheme conceals g^T and T from all entities by the same proof as the system secret S . The reason why $g^T \bmod p$ must be concealed is as follows: suppose that an entity knows $g^T \bmod p$ and broadcast keys U_1 and U_2 . Then, he can get another broadcast key U_3 where the corresponding preparation data $X_3 = X_1 \times X_2$, by calculating $U_3 = X_1^S \times X_2^S \times g^T \bmod p$, wherein $X_1^S = U_1/g^T \bmod p$ and $X_2^S = U_2/g^T \bmod p$.

Finally, we consider an attack to modify and forge the broadcast data. The key distributor can add the message recovery signature into the broadcast data without increasing the amount of transmission data. Then, the valid entities do not share the forged the broadcast key even if an attacker modifies and forges the broadcast data. Moreover, we consider that a time-stamp on the broadcast data is necessary to prevent a replay attack.

7 Other Considerations

New entity: When a new entity wants to join the broadcast group, the key distributor decides his unique identity number c which has not been used yet as the entity identification nor spare one. The key distributor calculates his secret key using $F(c)$ and $G(c)$, and sends it and the present broadcast key to him in a secure manner. If a previously excluded entity wants to join again, the key distributor issues a new identity number and a new secret key. This procedure does not affect the existing entities.

How to decide a parameter k : The key distributor can exclude $k - 1$ entities for one time at the maximum. And, the parameter k determines the amount of broadcast data. Moreover, k determines the lifetime of the broadcast key against the conspiracy. Because $k + 1$ entities with the conspiracy can obtain the system secret S , the key distributor shall re-set the system parameters before the total number of excluded entities becomes over $k + 1$. Therefore, the key distributor should decide the parameter k to fit for an actual system.

8 Evaluation

First, we compare our scheme with the previous scheme using a simple binary balanced tree which was proposed in [12] [13], and is recommended in RFC2627. Here, we suppose that the key distributor excludes only one entity, i.e., $d = 1$, $k = 2$. We suppose that the previous scheme uses 128 bit block cipher. Moreover, we suppose that our proposed scheme uses 160 bit elliptic curve cryptography and the symmetric cipher $E(.)$ of which output data size is the same as input data size.

Fig.1 shows that the data amount (bits) transmitted by the key distributor of our scheme does not relay on the number of total entities n . Therefore, our scheme is effective to achieve a quick key distribution with the exclusion when n is large. On the other hand, the transmission amount in the previous scheme increases as the increase of n . The result of the previous scheme in Fig.1 is derived from a table in [12].

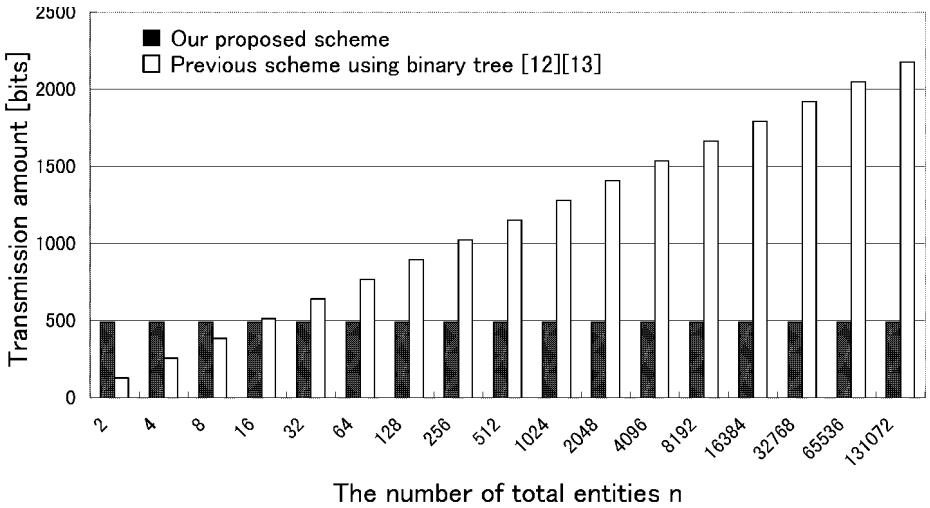


Fig. 1. Comparison of transmission amount between our scheme and the previous scheme.

Fig.2 shows that the key storage of each entity in our scheme does not relay on the number of total entities n . Therefore, our scheme is suitable for devices with a small memory. On the other hand, the key storage of the previous scheme increases to a large amount.

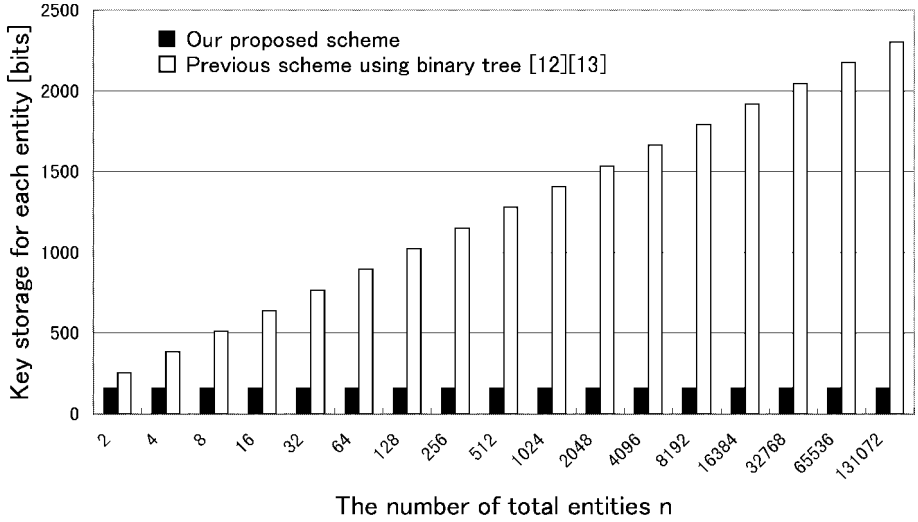


Fig. 2. Comparison of key storage between our scheme and the previous scheme.

Next, we compare our scheme with Anzai-Matsuzaki-Matsumoto scheme [1] in Table 1. In Table 1, $|p|$ denotes bit length of the parameter p , and so on. We add explanations for each of the comparison items as follows:

The number of exponentiations: In our scheme, each entity can calculate the broadcast key with only two modular exponentiations regardless of n and k . Since $2 \leq k$ holds, our scheme is always faster and requires less computing power than Anzai-Matsuzaki-Matsumoto scheme.

The amount of exclusion data: In our scheme, the key distributor broadcasts k exclusion data, similar to Anzai-Matsuzaki-Matsumoto scheme. The bit length of the exclusion data for each transmission is $|q|$ bits, while bits length of Anzai-Matsuzaki-Matsumoto scheme is $|p|$ bits. Therefore, the total amount of transmission data in our scheme can be reduced from Anzai-Matsuzaki-Matsumoto scheme (where the exclusion data has $|p|$ bits) when $|p| > |q|$ (e.g., over a prime field $\mathbf{Z}_p$).

The key storage of each entity: Our scheme requires each entity to store a small fixed amount of secret key, regardless of n and k . Though the storage size of our scheme is slightly more than one of Anzai-Matsuzaki-Matsumoto scheme, it can be reduced using an elliptic curve cryptography.

Who can decide the excluded entities: Anzai-Matsuzaki-Matsumoto scheme enables any entity to exclude any entity. On the other hand, only a key distributor can make the broadcast exclusion in our scheme. We consider that this assumption is reasonable in some actual applications.

Table 1. Comparison between our scheme and Anzai-Matsuzaki-Matsumoto scheme.

Comparison Items	Our proposed scheme	Anzai-Matsuzaki-Matsumoto scheme
The number of exponentiations (times)	2	$k + 1$
The amount of exclusion data (bits)	$k \times q $	$k \times p $
The key storage of each entity (bits)	$ p + q $	$ q $
Who can decide the excluded entities	Only key distributor	Any entity

9 Conclusion

In this paper, we have proposed a scheme for the broadcast exclusion, in which each entity computes the broadcast key with two modular exponentiations. This allows devices with low computing power such as a telecommunications terminal or a smart card to obtain the broadcast key in a reasonable time.

References

1. J.Anzai, N.Matsuzaki, T.Matsumoto, "A Quick Group Key Distribution Scheme with Entity Revocation," *Proceedings of ASIACRYPT'99*, LNCS1716, pp. 333-347, Springer-Verlag, 1999.
2. C.Blundo, L.A.Fronta Mattos, and D.R.Stinson, "Generalized Beimel-Chor schemes for broadcast encryption and interactive key distribution," *Theoretical Computer Science*, 200(1-2):pp. 313-334, 1998.
3. M.Burmester, Yvo.Desmedt, "A Secure and Efficient Conference Key Distribution System," *Proceedings of EUROCRYPT'94*, LNCS950, pp.275-285, Springer-Verlag, 1994.
4. R.Canetti, J.Garay, G.Itkis, D.Micciancio, M.Naor, and B.Pinkas, "Multicast Security: A Taxonomy and Efficient Constructions," *Proc. INFOCOM'99*, Vol.2, pp. 708-716, March 1999.
5. R.Canetti, T.Malkin and K.Nissim, "Efficient Communication-storage Tradeoffs for Multicast Encryption," *Proceedings of EUROCRYPT'99*, LNCS1592, pp.459-474, Springer-Verlag, 1999.
6. Y.Desmedt, Y.Frankel, "Threshold Cryptosystems," *Proceedings of CRYPTO'89*, LNCS435, pp.307-315, Springer-Verlag, 1989.
7. A.Fiat and M.Naor, "Broadcast Encryption," *Proceedings of CRYPTO'93*, LNCS773, pp.480-491, Springer-Verlag, 1993.
8. R.Kumar, S.Rajagopalan and A.Sahai, "Coding Constructions for Blacklisting Problems without Computational Assumptions," *Proceedings of CRYPTO'99*, LNCS1666, pp.609-623, Springer-Verlag, 1999.
9. M.Luby and J.Staddon, "Combinatorial Bounds for Broadcast Encryption," *Proceedings of EUROCRYPT'98*, LNCS1403, pp. 512-526, Springer-Verlag, 1998.

10. K.Nyberg, R.A.Rueppel, "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem," *Proceedings of EUROCRYPT'94*, LNCS950, pp. 182-193, 1995.
11. A.Shamir, "How to Share a Secret," *Comm.Assoc. Comput. Mach.*, vol. 22, no. 11, pp. 612-3, Nov. 1979.
12. D.M.Wallner, E.J.Harder and R.C.Agee, "Key Management for Multicast: Issues and Architectures," Internet RFC2627, <ftp://ietf.org/rfc/rfc2627.txt>. June, 1999.
13. C.K.Wong, M.Gouda and S.S.Lam, "Secure Group Communications Using Key Graphs," *Proceedings of ACM SIGCOMM'98*, 1998. Also, Technical Report TR 97-23, Department of Computer Sciences, the University of Texas at Austin, July 1997.

Cheating Prevention in Secret Sharing

Hossein Ghodosi¹ and Josef Pieprzyk²

¹ School of Information Technology, James Cook University
Townsville Qld 4811, Australia
`hossein@cs.jcu.edu.au`

² School of IT and CS, University of Wollongong
Wollongong NSW 2522, Australia
`josef@cs.uow.edu.au`

Abstract. The commonly used technique for cheating detection requires that extra information be given to each participant. However, in a secret sharing scheme when the size of shares increases the security of the system degrades. In this paper we study the cheating problem in Shamir's scheme (in the sense of Tompa and Woll [1] attack) and present alternative solutions to this problem. First we consider cheating prevention via longer shares. Next we introduce redundant secret sharing schemes and show how they can be used for cheating prevention. Nonlinear secret sharing offers also some protection against cheaters. The work concludes with a discussion about a combined approach.

1 Introduction

Since its invention, secret sharing has become an indispensable tool for group cryptography. The concept of secret sharing was independently formulated by Blakley [3], Chaum [4] and Shamir [5]. Secret sharing is set up by a trusted party also called the dealer. Knowing a secret, the dealer creates shares of the secret and distributes them to the participants (also called shareholders). After that the dealer disappears and the secret is collectively held by the group. To recover the secret, a big enough collection of participants pools their shares together and recreates the secret. The recovery of the secret can be done collectively when all active participants reveal their shares to each other or the participants delegate a trusted party called the combiner.

Unfortunately, the recovery of the secret can be easily corrupted by a dishonest participant who pools a modified share (instead of the original). The secret recovered by the combiner is obviously different from the original but the dishonest participant can compute the original secret, leaving other honest principals with an invalid secret. This way of cheating was discussed by Tompa and Woll [1]. Also they suggested a method to prevent the cheating. Since that time, there has been a considerable effort to investigate ways of cheating prevention.

2 Previous Works

Cheating prevention can be considered in either unconditionally or conditionally secure secret sharing. In unconditionally secure secret sharing, cheaters are identified by the combiner by checking the validity of their shares. It is assumed that once all shares have passed through validation, the recovered secret must be valid as well. Rabin and Ben-Or [9] showed how to validate shares by checking whether or not they satisfy a system of linear equations. Carpentieri [10] demonstrated a similar scheme but with shorter shares. The cheating prevention method suggested by Tompa and Woll [1] takes away the ability of a dishonest participant to calculate the valid secret from the invalid one. One straightforward solution to the problem of cheating is to have the trusted dealer sign each share prior to handing it out to a participant. Hence, a dishonest participant will be detected in the secret reconstruction phase. However, as pointed out in [1], all currently known signature schemes are conditionally secure, whereas the secret sharing scheme under investigation (the Shamir scheme) is an unconditionally secure scheme.

In conditionally secure secret sharing, there are two related problems which have been addressed: (non-interactive) share verification and secret verification. Feldman [6] suggested a solution for the Shamir scheme. While setting up the scheme, the dealer announces check values g^{a_i} where a_i are co-efficients of the Shamir polynomial $f(x) = a_0 + a_1x + \dots, a_{t-1}x^{t-1}$ and g is a primitive element of the cyclic group in $GF(p)$. The prime p must be large enough so the instances of the discrete logarithm problem are intractable. Pedersen [7] designed a verifiable secret sharing based on a commitment scheme which follows Feldman's idea of public check values. The public checks can be used in two different circumstances when participants want to verify the validity of their shares obtained from the dealer or when the combiner wishes to verify the shares. A class of publicly verifiable secret sharing was introduced by Stadler in [8].

This work is structured as follows. The next Section describes basic definitions and notions used in secret sharing together with the Tompa-Woll (T-W) attack. Section 4 investigates the cheating success when shares given to participants are lengthened. Applicability of redundant secret sharing is studied in Section 5. Section 6 defines nonlinear secret sharing and examines its cheating prevention characteristics. The work is closed by Section 7 which discusses joint application of previously described cheating prevention methods.

3 Basic Concepts of Secret Sharing

In secret sharing, the basic active entity is a group $\mathcal{P}$ of n participants $\mathcal{P} = \{P_1, \dots, P_n\}$ who collectively holds a secret. The access structure Γ of the group $\mathcal{P}$ is the collection of all subsets who are authorized to jointly recover the secret. A (t, n) threshold scheme allows the secret to be recovered if the currently active subgroup $\mathcal{A} \subset \mathcal{P}$ consists of t or more participants, or

$$\Gamma = \{\mathcal{A} \mid \#\mathcal{A} \geq t\}$$

where $\#\mathcal{A}$ means the cardinality of $\mathcal{A}$. Secret sharing includes two generic algorithms

1. *the dealer* which sets up the scheme, i.e. takes a secret to be shared and splits it into shares which are securely conveyed to participants,
2. *the combiner* which reconstructs the secret. A currently active participants from the group $\mathcal{A}$ send their shares via secure channels. The combiner is always successful if the currently active group $\mathcal{A}$ belongs to the access structure Γ . Otherwise if $\mathcal{A} \notin \Gamma$, it fails with an overwhelming probability.

A secret sharing is called *perfect* if any unauthorized sub-group of participants cannot learn anything about the secret. More precisely

$$H(K \mid \mathcal{A} \notin \Gamma) = H(K)$$

where $H(K)$ is the entropy of the secret K . Clearly, $H(K \mid \mathcal{A} \in \Gamma) = 0$. A perfect secret sharing is called *ideal* if the size of shares is equal to the size of the secret.

The Shamir (t, n) threshold scheme is based on polynomial interpolation. Given t points on the two-dimensional plane $(x_1, y_1), \dots, (x_t, y_t)$ with distinct x_i 's, there is the unique polynomial $f(x)$ of degree at most $t - 1$ such that $y_i = f(x_i)$ for all i . The Lagrange interpolation allows the recreation of the polynomial using the following expression

$$f(x) = \sum_{i=1}^t y_i \prod_{\substack{j=1 \\ j \neq i}}^t \frac{x - x_j}{x_i - x_j}. \quad (1)$$

Let the secret be an element of a finite field, that is, $K \in GF(p)$, where p is a prime.

Dealer

1. chooses n distinct and non-zero elements of $\mathbb{Z}_p$, denoted $x_1, \dots, x_n$ and publishes them together with the assignment of x_i to each P_i .
2. secretly picks up (independently at random) $t - 1$ elements of $\mathbb{Z}_p$, denoted $a_1, \dots, a_{t-1}$ and forms the polynomial

$$f(x) = K + \sum_{i=1}^{t-1} a_i x^i.$$

3. For $1 \leq i \leq n$, computes s_i such that

$$s_i = f(x_i) \pmod{p}.$$

4. gives (in private) share s_i to participant P_i .

Combiner

1. Every collection of at least t participants can apply the Lagrange interpolation and reconstruct the polynomial $f(x)$ and hence recover the secret $K = f(0)$. An alternative method of secret reconstruction may use a system of linear equations for which the unknown K is sought.

It is well-known that the Shamir scheme is perfect. That is, the probability of an unauthorized set of participants being able to determine the secret is no better than that of an outsider, and therefore is no better than guessing the secret. It is also clear that Shamir's scheme is ideal, that is, the length of the share assigned to each participant is the same as the length of the secret.

3.1 Tompa-Woll Attack on Shamir Schemes

Tompa and Woll [1] discovered a way dishonest participants can cheat when using the (t, n) Shamir secret sharing based on the polynomial $f(x)$. Each participant holds her share $s_i = f(x_i)$ and x_i is public; $i = 1, \dots, n$. Without the loss of generality, assume that the currently active sub-group is $\mathcal{A} = \{P_1, \dots, P_t\}$. Let the collection $\mathcal{C} = \{P_1, \dots, P_\ell\}$ be cheaters who collectively wish to fool the other active participants (clearly $\mathcal{C} \subset \mathcal{A}$). If the cheaters know the currently active sub-group (note that this is an important assumption), then they can collectively define a random polynomial $\Delta(x)$ such that

$$\Delta(x_i) = \begin{cases} 0 & \text{for } P_i \in \mathcal{A} \setminus \mathcal{C} \\ \delta_i & \text{for } P_i \in \mathcal{C} \end{cases}$$

where δ_i are chosen by $\mathcal{C}$. We assume that $\Delta(0) \neq 0$. At the pooling time, honest participants submit their shares s_i , $i = \ell + 1, \dots, t$ while cheaters give $s_i + \delta_i$. The combiner uses the Lagrange interpolation and recovers the polynomial $f(x) + \Delta(x)$ and announces the wrong secret $K' = f(0) + \Delta(0)$. The cheaters know the polynomial $\Delta(x)$ and the value $\Delta(0)$ so they can recover the secret $K = f(0) = K' - \Delta(0)$.

The polynomial $\Delta(x)$ is likely to be of degree $t - 1$ if the cheaters select their δ_i at random. The other extreme is the deterministic option when

$$\Delta'(x) = \prod_{i=\ell+1}^t (x - x_i)$$

of degree $t - \ell$. The values of $\delta_i = \Delta'(x_i)$ for $i = 1, \dots, \ell$. This option may not be preferred by cheaters as the value $\Delta'(0)$ is public and honest participants are likely to try it after a corrupted key recovery. Note that a better option for cheaters is to generate one δ at random, say δ_1 , and use the Lagrange interpolation to the points $((x_1, \delta_1), (x_{\ell+1}, 0), \dots, (x_t, 0))$. The resulting polynomial $\Delta(x)$ is of degree $t - \ell + 1$ and the value $\Delta(0)$ is known to the cheaters only (the uncertainty of the $\Delta(0)$ and the secret are the same).

Tompa and Woll observed that if the co-ordinates x_i are kept secret, then the cheaters cannot design the requested $\Delta(x)$ polynomial. So they suggest to select x_i s uniformly and randomly from all possible values in $GF(p)$, where the prime module p is

$$p > \max((|\mathcal{K}| - 1)(t - 1)/\varepsilon + t, n).$$

The share of participant P_i is a pair $(x_i, f(x_i))$. They have shown that with this modification to the Shamir scheme, the possibility of successful (undetected) cheating is less than ε ($\varepsilon > 0$). The mechanism for cheating detection is that not every element of $GF(p)$ is now a legal secret. Hence, the probability of cheating and recovering one of the $|\mathcal{K}| - 1$ legal but incorrect secret is less than ε .

Although cheaters are detected with high probability, they can obtain the secret while the other participants gain no information about the secret. In order to avoid this undesirable situation, the dealer sets up a sequence of schemes such that only one of them is associated with the secret K and the other ones are associated with an illegal secret, $S \in GF(p) - \mathcal{K}$ (S is known to all participants). In secret construction phase, when t participants agree to pool their shares, they reconstruct the secrets one at a time, until the recovered secret is not S . This terminates the protocol. If the reconstructed secret is not legal, then cheating has occurred.

Carpentieri, Santis, and Vacaro [2] considered the cheating problem in the sense of the Tompa and Woll attack and showed that the phenomenon of increasing the size of the shares is unavoidable. They demonstrated that in any secret sharing scheme that has probability of successful cheating less than $\varepsilon > 0$, it must give to each participant shares at least the size of the secret plus $\log \frac{1}{\varepsilon}$. Note that the size of shares, $\log |s_i|$, in Tompa and Woll's scheme is

$$2 \log \left(\frac{(|\mathcal{K}| - 1)(t - 1)}{\varepsilon} + t \right) < \log |s_i| < 2 \log \left(\frac{(|\mathcal{K}| - 1)(t - 1)}{\varepsilon} + t \right) + 1.$$

4 Cheating Prevention via Longer Shares

Tompa and Woll's attack works because for every set of t points $(x_1, y_1), \dots, (x_t, y_t)$ with distinct x_i s there is a unique polynomial $f(x)$ of degree at most $t - 1$ such that $y_i = f(x_i)$ for all i . That is, every set of t such values can be selected randomly and independently (if the secret is not given and fixed). Since the cooperating participants cannot verify the validity of the secret they have to accept it. However, the correctness of the secret can be verified if an extra share of the original secret is available. So if a cheater modifies her share, there is a high probability that the resulting polynomial will not pass through points of honest participants (if those points were not used in the Lagrange approximation). So we can use the following mechanism for cheating detection in Shamir's scheme.

Lemma 1. *In a (t, n) Shamir scheme, if $t+1$ participants cooperate in the secret reconstruction phase then the probability of successful Tompa-Woll attack is $\frac{1}{p}$.*

Proof. If there is cheating in the system, then t_1 points are independent and random. Therefore, with high probability, they will reconstruct a polynomial of degree t , which determines cheating. The probability of recovering a polynomial of degree $t - 1$ is equivalent to the probability of constructing Shamir $(t + 1, n)$ scheme and having a polynomial with degree less than t (which is equal to $\frac{1}{p}$).



Note that this mechanism may not be applicable if an extra share (participant) is not available (for example, in a (n, n) scheme). However, this is not the case with cheating detection schemes, in which the system allows every participant to be given extra information. In the following we show how this mechanism can be used to solve the cheating problem in the Shamir threshold scheme.

Let p be large enough such that the dealer can generate the desirable number of shares. The number of shares, as we will see shortly, depends on the required probability for cheating detection.

Dealer

- chooses $2n$ distinct and non-zero elements of $\mathbb{Z}_p$, denoted $x_1, \dots, x_{2n}$ and sends x_i, x_{2i} to P_i via a public channel.
- secretly selects (independently at random) $2t - 2$ elements of $\mathbb{Z}_p$, denoted $a_1, \dots, a_{2t-2}$ and forms the polynomial

$$f(x) = K + \sum_{i=1}^{2t-2} a_i x^i.$$

- for $1 \leq i \leq 2n$, computes s_i , where

$$s_i = f(x_i) \pmod{p}.$$

- gives (in private) shares s_i and s_{2i} to participant P_i .

That is, the dealer constructs a $(2t - 1, 2n)$ Shamir's scheme. Hence, in the secret reconstruction phase, every set of at least t participants (since each participant has two shares) can apply the Lagrange interpolation formula to reconstruct the polynomial and hence to recover the secret.

When t participants collaborate, they know $2t$ shares. However, $2t - 1$ shares is sufficient to reconstruct the original polynomial. If there is cheating then with the probability $\frac{1}{p}$, the extra shares will be satisfied with the constructed polynomial. That is, no matter how many cheaters are in the group, the probability of successful cheating is $\frac{1}{p}$.

If we want to decrease the probability of successful cheating then the dealer constructs a $(qt - (q - 1), qn)$ Shamir scheme and sends the shares to corresponding participants. That is, participant P_i receives the tuple $(s_i, s_{2i}, \dots, s_{qi})$, where shares are generated as in the original Shamir scheme. Similar to the above scheme, clearly, every t participant can cooperate in order to recover the secret. If every participant contributes with q shares and the resulting polynomial is of degree $qt - (q - 2)$ then, with probability $\frac{1}{p^{q-1}}$ there is no cheating in the system. Otherwise, the scheme detects cheating.

It is easy to see that for any extra share given to each participant the probability of successful cheating is decreased by $\frac{1}{p}$. However, the size of the shares in this scheme is $\log p$. That is, in order to have the probability of successful cheating less than $\frac{1}{p}$ we give each participant his share plus extra shares of size $\log p$. Thus, our scheme meets the best theoretical result given in [2].

An alternative method is to increase the size of p (this is also necessary if p is not large enough so that the dealer can generate as many shares as he wishes). If we want to give each participant q extra shares the probability of successful cheating is $\frac{1}{p^{q-1}}$. The prime module $p > qt$ is sufficient to construct such a system.

The construction of the scheme and the probability of cheating detection is similar to the case discussed before. Note that although now the size of each share is slightly larger than the secret, the probability of cheating is smaller than what we have discussed in the previous scheme.

5 Cheating Protection via Redundant Secret Sharing

We explore how the idea of “hiding” the secret sharing among other secret sharing, can be used for both cheating detection and cheater identification.

Given two secret sharing schemes S_1 and S_2 with their corresponding access structures Γ_1 and Γ_2 , respectively, and $\Gamma_1, \Gamma_2 \subset \mathcal{P}$. Secret sharing S_i holds the secret K_i , $i = 1, 2$. The *Cartesian product* of S_1 and S_2 is a scheme denoted as $S = S_1 \times S_2$ which holds the secret $K = (K_1, K_2)$. Given active group $\mathcal{A}$ of participants. The secret key K can be recovered only if $\mathcal{A} \in \Gamma_1 \cap \Gamma_2$. Clearly partial secret K_i can be recovered if $\mathcal{A} \in \Gamma_i$. The secret is unknown if $\mathcal{A} \notin \Gamma_1$ and $\mathcal{A} \notin \Gamma_2$.

Given two Shamir secret sharing schemes S_1, S_2 where S_i is a (t_i, n) threshold scheme based on the polynomial $f_i(x)$ over $GF(p_i)$, $i = 1, 2$. Then $S = (S_1, S_2)$ is a secret sharing with the pair of polynomials $(f_1(x), f_2(x))$. The secret is $K = (K_1, K_2)$. Two secret sharings are *independent* if they were generated independently of each other.

Lemma 2. *Given a secret sharing $S = S_1 \times S_2$ where S_1 and S_2 are independent Shamir secret sharing schemes defined as above. Then the scheme S is*

- *ideal and perfect if and only if $t_1 = t_2$,*
- *is a ramp scheme (t_1, t_2, n) for $t_1 < t_2$ and*
- *is a ramp scheme (t_2, t_1, n) for $t_2 < t_1$.*

The proof is obvious and skipped. The definition of the ramp scheme can be found in [11].

Now we are ready to define our first scheme. Given two independent Shamir secret sharing schemes S_1 and S_2 . S_1 is based on a polynomial $f(x)$ and S_2 is based on $-g(x)$. Both polynomials are computed over $GF(p)$.

Dealer. Given secret $K \in GF(p)$ to be collectively held by the group $\mathcal{P}$. The dealer sets up a Shamir scheme S_1 for the secret with the polynomials $f(x)$ of

degree precisely $(t-1)$. Next the dealer sets up S_2 over $GF(p)$ with a polynomial $g(x)$ of degree precisely $(\ell-1)$ for a random secret k . Shares $s_i = (\alpha_i, \beta_i) = (f(x_i), g(x_i))$ are secretly communicated to participants.

Combiner. Assume that the active group is $\mathcal{A} = \{P_1, \dots, P_t\}$. The members of $\mathcal{A}$ submit their shares to the combiner via secure channels. The combiner recreates both polynomials $f(x)$ and $g(x)$ using the Lagrange interpolation. Now it tests the degree of polynomials and identifies $f(x)$ and returns the secret $K = f(0)$ to active participants via secure channels.

The scheme has the following properties:

- is not perfect (the degree of $f(x)$ must be $t-1$ so any collection of $(t-1)$ principals is able to exclude a single value from the range of possibilities for the secret),
- is not ideal (the share is twice as long as is necessary).

The ability for cheating prevention is discussed by the following lemma.

Lemma 3. *Given a secret sharing S which is a Cartesian product of two Shamir schemes based on $(f(x), g(x))$ defined as above. Then the combiner can detect T-W cheating with the probability $\frac{1}{2}$ of any coalition of cheaters $\mathcal{C} = \{P_1, \dots, P_\ell\}$ only if $\ell \leq \frac{t+1}{2}$.*

Proof. The cheaters can pool together their shares and they are unable to identify shares as both collections are statistically indistinguishable. The cheating can only be detected if the minimal required degree of $\Delta(x)$ is bigger than the degree of $g(x)$. The minimal degree is $t-\ell$ and at the same time the degree of $g(x) = \ell-1$ so solving $t-\ell > \ell-1$ gives us the requested condition on ℓ . If ℓ is bigger then the degrees of both $\Delta(x)$ and $g(x)$ can be of the same degree and the cheaters can successfully launch the T-W attack.

If $\ell \leq \frac{t+1}{2}$, the cheaters have three possibilities:

- not to cheat; in this case the combiner will work correctly,
- apply the T-W attack to one polynomial. If they have targeted $f(x)$, they succeed. Otherwise, if $g(x)$ is modified then the combiner will recover $g(x) + \Delta(x)$. From our previous considerations, we know that the degree of $\Delta(x)$ must be at least $t-\ell$ so consequently the degree of $g(x) + \Delta(x)$ is also $t-\ell$ and this fact will be detected by the combiner.
- apply the T-W attack to both polynomials. This time the modification of $g(x)$ will always be detected. It is reasonable to assume that the cheaters will disregard this option.

So the conclusion follows. ♠

Corollary 1. *The above scheme based on $(f(x), g(x))$ allows cheaters to be identified if $\ell \leq \frac{t+1}{2}$ and $\ell < t-\ell$ with the probability $\frac{1}{2}$.*

Proof. Once the coalition $\mathcal{C}$ of cheaters have modified $g(x)$, the combiner knows ℓ corrupted shares and $\ell + u$ correct ones $u \geq 1$ and $t = 2\ell + u$. Knowing that $g(x)$ is of degree $\ell - 1$, the combiner may try to enumerate all $\binom{t}{\ell}$ possible polynomials. Among them, there must be a cluster of $\ell + u$ repetitions identifying uncorrupted shares (and participants). ♠

The discussed cheating prevention gives a relatively high probability of success to cheaters. To reduce the probability of success, we can use many controlled secret sharings. Given our secret sharing S_1 with the polynomial $f(x)$ and the secret $K = f(0)$. We design a sequence of Shamir schemes $S_2, \dots, S_r$ each based on the corresponding polynomial $g_i(x)$; $i = 2, \dots, r$. All secret sharing schemes are constructed over the same $GF(p)$. Now we build the Cartesian product $S = S_1 \times \dots \times S_r$. Clearly, if we make the previous assumption as to the size ℓ of coalition of cheaters, then the scheme S allows cheaters to be detected and identified with the probability $1 - \frac{1}{r+1}$.

6 Nonlinear Secret Sharing

Obviously, the T-W attack works because the Shamir scheme is linear. A way to prevent this could be an introduction of secret sharing which is nonlinear.

Given a group $\mathcal{P}$ of n participants. Assume that we have already a Shamir secret sharing based on a polynomial $f(x) \in GF(p^t)$ which allows any t to recover the secret $K = f(0)$. Note that the polynomial $f(x)$ can be treated as an element of $GF(p^t)$. We would like to apply a nonlinear permutation $\Pi : GF(p^t) \rightarrow GF(p^t)$ on the element $f(x)$. We have the following possibilities

- to choose a random permutation,
- to design a nonlinear permutation.

We assume that we choose a random permutation Π . The permutation is public. The randomized Shamir scheme is defined as follows.

Dealer sets up the scheme and

- designs a (t, n) Shamir scheme with the polynomial $f(x) \in GF(p^t)$,
- computes $F(x) = \Pi(f(x))$ in $GF(p^t)$.
- distributes via secure channel shares $s_i = F(x_i)$ to participants. Each participant holds her pair (x_i, s_i) . Shares s_i are private while x_i are public.

Combiner is activated by a sub-group of t participants and

- collects t shares from participants,
- computes the polynomial $F(x)$ using Lagrange interpolation,
- recovers the Shamir polynomial $f(x) = \Pi^{-1}(F(x))$ in $GF(p^t)$ and the secret $K = f(0)$,
- the secret $K \in GF(p)$ is communicated privately to all active participants.

Consider the group $\mathcal{P}$ collectively knows the polynomial $F(x) \in GF(p^t)$. Note that in (t, n) threshold scheme, any collection of t participant holds the unique polynomial $F(x)$. Any $(t-1)$ participants, say $\{P_1, \dots, P_{t-1}\}$, hold a set $\mathcal{F}$ of p different polynomials among which there is the correct one $F(x)$ where

$$\mathcal{F} = \{g(x) | g(x_i) = s_i; i = 1, \dots, t-1, g(x_t) \in GF(p)\}.$$

Lemma 4. *The randomized secret sharing is perfect only if the set*

$$\{g(0) | g(x) \in \Pi^{-1}(\mathcal{F})\}$$

is of cardinality p (consists of all different elements) for any active set of $(t-1)$ participants.

The proof is obvious. Note that for a random Π , perfectness is a rare event as it requires that for all polynomials from $\mathcal{F}$ their preimages are polynomials with different constants.

6.1 Modified Tompa-Woll Attack

We are going to describe a version for $(t-1)$ cheaters. The general case of the attack can be easily derived. Assume that the set of currently active participants is $\mathcal{A} = \{P_1, \dots, P_t\}$ with the coalition of cheaters $\mathcal{C} = \{P_1, \dots, P_{t-1}\}$. The attack works for unconditional setting where the attackers have unlimited computing resources. The attack progresses as follows.

- Cheaters collectively decide on a polynomial $\Delta(x) \in GF(p^t)$ such that $\Delta(x_t) = 0$ ($\Delta(x) \neq F(x)$).
- Shares of cheaters (generated from the polynomial $\Delta(x)$) are submitted to the combiner. The honest participant P_t submits her share s_t .
- The combiner takes all shares and recovers a polynomial $g'(x)$, transforms it $f'(x) = \Pi^{-1}(g'(x))$ and returns the constant $K' = f'(0)$.
- Cheaters now exhaustively search through all polynomials from the set

$$\mathcal{F} = \{g(x) | g(x_i) = s_i; i = 1, \dots, t-1, g(x_t) \in GF(p)\},$$

find the set $\Pi^{-1}(\mathcal{F})$ and its subset

$$\mathcal{G}(K') = \{g(x) | g(x) \in \Pi^{-1}(\mathcal{F}) \text{ and } g(0) = K'\}.$$

- The cheaters may now determine a set of elements

$$\mathcal{S}_t = \{g(x_t) | g(x) \in \Pi(\mathcal{G})\}$$

The set $\mathcal{S}_t$ must include the share of the honest participant P_t .

- For each element from $\mathcal{S}_t$, the cheaters can compute the all possibilities for the true secret K .

To maximize the chance of success, the cheaters can run through all possible sets defined by vectors $(a_1, \dots, a_{t-1})$; $a_i \in GF(p)$

$$\mathcal{F}(a_1, \dots, a_{t-1}) = \{g(x) | g(x_i) = a_i; i = 1, \dots, t-1 \text{ and } g(x_t) \in GF(p)\}.$$

The set $\Pi^{-1}(\mathcal{F}(a_1, \dots, a_{t-1}))$ can be split into subsets $\mathcal{G}(k) = \{g(x) | g(x) \in \Pi^{-1}(\mathcal{F}) \text{ and } g(0) = k\}$ where $k \in GF(p)$. The cardinality of the biggest subset $\mathcal{G}(k)$ gives the lower bound on the probability of a successful cheating.

It is interesting to observe that to get the randomized secret sharing perfect, one needs to make sure that the random permutation Π^{-1} translates in a one-to-one fashion constants in candidates for $f(x)$ into constants of candidates for $F(x)$. This is apparently at odds with the requirements for the scheme to be resistant against the T-W attack. Is it possible to find a compromise in which perfectness is traded off with the resistance against the T-W attack ?

6.2 Unconditional Security

It should be clear by now that if the secret sharing is to be used in the unconditionally secure setting, then random selection of the permutation Π is not a very good idea. It is desirable to select Π very carefully, perhaps, using combinatorial structures whose properties are well known. What properties are to be required from a “well designed” permutation Π ?

Let us enumerate the ones we have identified so far.

1. Π is a permutation over $GF(p^t)$.
2. For an arbitrary polynomial $\Delta(x)$, the preimage of the set $\mathcal{F} = \{g(x) | g(x) = \Delta(x) + a; a \in GF(p)\}$ or

$$\Pi^{-1}(\mathcal{F})$$

should have polynomials with the same constants. Note that if $\Pi^{-1}(\mathcal{F})$ contains polynomials with all different constants the cheaters will always succeed. The scheme is then perfect.

6.3 Conditional Security

These considerations are valid if the secret belongs to a large space so the cheaters cannot exhaustively search the space. Encryption algorithms provide a sufficiently large collection of permutations conveniently indexed by their (private or public) keys. Given a private-key encryption algorithm such as DES, then one can design permutations whose input/output size is a multiple of the block size of the algorithm (using the classical permutation-substitution network). The T-W attack will proceed as before with the important difference that cheaters will be able to find preimages of $\Delta + a$ where a belongs to a small subset whose size is related to the computational power of cheaters. If the combiner returns a fake secret then the cheaters may succeed with some probability proportional to the size of the subset.

Consider the case when the permutation is a public key encryption. Given a generic public-key cryptosystem with two algorithms: public encryption E and private decryption D . Both are permutations (such as the RSA). A (t, n) secret sharing based on E, D is defined below.

Dealer

- constructs a Shamir scheme with polynomial $f(x) \in GF(p^t)$,
- builds an instance of public key cryptosystem with two algorithms E, D . Note that the message/ciphertext domain must be a superset of $GF(p^t)$. Publishes E algorithm,
- converts $f(x)$ into $F(x)$ or

$$F(x) = D(f(x)) \in GF(q^t)$$

where $q \geq p$,

- generates shares $s_i = F(x_i)$ and distributes secretly to participants.
- dies (and forgets the decryption algorithm).

Combiner

- accepts shares from active participants. Having received at least t , it is able to recover $F(x)s$,
- uses the public encryption algorithm to obtain $f(x) = E(F(x))$,
- returns $K = f(0)$ to all active participants via secure channels.

Theorem 1. *Given the above defined secret sharing with the slight modification that the combiner returns the recovered polynomial $f(x)$ instead of $K = f(0)$. A collection of $(t - 1)$ cheaters can fool the honest participant if and only if finding the private decryption algorithm is “easy”.*

Proof. (Sketch) We are going to show how the (private) decryption algorithm can be constructed from an algorithm C which is used by the successful cheaters. The C algorithm is a probabilistic one which accepts that a random polynomial $\Delta(x)$ ($\Delta(x)$ is also denoted by Δ if we treat it as an integer) and $f'(x)$ (or f' for short) and exits the share of the honest participant (or equivalently the secret).

Given a cryptogram $c = E(m)$ where m is the message. Note that the encryption algorithm can be seen as the trusted combiner who takes the random Δ and the secret share of the honest participant $m - \Delta$ and returns the prescribed $c = E(m)$. Now we use the algorithm C . We input Δ and c to it and collect the share of the honest participant which is $m - \Delta$. This obviously allows the encryption algorithm to be reversed. This establishes the equivalence of the two security aspects. ♠

7 Combined Approach

Separate applications of Cartesian products and nonlinear secret sharing offer some benefits but certainly, the application of the two measures jointly may

offer far better protection against cheaters. Our exposition is simplified as we are going to use a single redundant secret sharing as the generalization is trivial (but unnecessarily complicated). Clearly, the secret sharing allows any t out of n to recover the secret.

Dealer

- designs a pair of (t, n) Shamir schemes with the underlying polynomials $f(x), g(x) \in GF(p^t)$ where the secret is $K = f(0)$ and $g(x)$ is the redundant polynomial which can be easily identified by the combiner. How? This will be discussed later.
- selects a publicly accessible permutation $\Pi : GF(p^{2t}) \rightarrow GF(p^{2t})$ and computes $F(x) = \Pi(f(x), g(x))$ where $(f(x), g(x)) \in GF(p^{2t})$ is a concatenation of the two polynomials,
- secretly distributes shares $s_i = F(x_i)$. Note that selection of x co-ordinates for polynomials f, g and F is public and it is reasonable to assume that x_i used for polynomials f, g and F are identical.

Combiner

- calls for shares and having t of them, recovers $F(x)$,
- applies the inverse permutation $\Pi^{-1}(F(x))$ and retrieves $(f(x), g(x))$,
- identifies which polynomial is redundant and returns the secret $K = f(0)$. If the identification fails, the combiner aborts.

This time selection of linear permutation Π is an option. Note that the coalition of $t - 1$ cheaters does not know the share of the honest participant which is twice the length of the secret. So their uncertainty about shares of the honest participants equals $H = \log_2 p^2$. After getting the secret recovered by the combiner, the cheaters reduce their uncertainty to $H = \log_2 p$ which is equal to the entropy of the secret.

One of the steps for the combiner is the identification of the redundant polynomial which gets rejected. Let us enumerate some of the options. The polynomial $g(x)$ can be

- fixed and publicly known (a single element from $GF(p^t)$),
- selected at random from all polynomials which contain $t - \ell$ public points (the space of such polynomials is of cardinality p^ℓ). If $\ell = t$ there are no public points and identification is impossible,
- chosen at random where $g(0) = h(K)$ where the function h is publicly accessible

Note that redundant $g(x)$ makes cheating slip through unnoticed while the fixed $g(x)$ enables the cheaters to extract more information about the secret (after getting the reply from the combiner).

References

1. M. Tompa and H. Woll, "How To Share a Secret with Cheaters," *Journal of Cryptology*, vol. 1, no. 2, pp. 133–138, 1988.

2. M. Carpentieri, A. Santis, and U. Vaccaro, "Size of Shares and Probability of Cheating in Threshold Schemes," in *Advances in Cryptology - Proceedings of EUROCRYPT '93* (T. Helleseeth, ed.), vol. 765 of *Lecture Notes in Computer Science*, pp. 118–125, Springer-Verlag, 1994.
3. G. Blakley, "Safeguarding cryptographic keys," in *Proceedings of AFIPS 1979 National Computer Conference*, vol. 48, pp. 313–317, 1979.
4. D. Chaum, "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups," tech. rep., Memorandum No. UCB/ERL M/79/10, University of California, Berkeley, CA, Feb. 1979.
5. A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, pp. 612–613, Nov. 1979.
6. P. Feldman, "A Practical Scheme for Non-interactive Verifiable Secret Sharing," in *28th IEEE Symposium on Foundations of Computer Science*, pp. 427–437, Oct. 1987.
7. T. Pedersen, "A Threshold Cryptosystem without a Trusted Party," in *Advances in Cryptology - Proceedings of EUROCRYPT '91* (D. Davies, ed.), vol. 547 of *Lecture Notes in Computer Science*, pp. 522–526, Springer-Verlag, 1991.
8. M. Stadler, "Publicly Verifiable Secret Sharing," in *Advances in Cryptology - Proceedings of EUROCRYPT '96* (U. Maurer, ed.), vol. 1070 of *Lecture Notes in Computer Science*, pp. 190–199, Springer-Verlag, 1996.
9. T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority" in *Proceedings of 21st ACM Symposium on Theory of Computing*, pp. 73–85, 1989.
10. M. Carpentieri. A perfect threshold secret sharing scheme to identify cheaters. *Designs, Codes and Cryptography*, 5(3):183–187, 1995.
11. G. Blakley and C. Meadows, "Security Of Ramp Schemes," in *Advances in Cryptology - Proceedings of CRYPTO '84* (G. Blakley and D. Chaum, eds.), vol. 196 of *Lecture Notes in Computer Science*, pp. 242–268, Springer-Verlag, 1985.

On Multiplicative Secret Sharing Schemes

Huaxiong Wang¹, Kwok Yan Lam², Guo-Zhen Xiao², and Huanhui Zhao²

¹ Centre for Computer Security Research
School of IT and CS
University of Wollongong
Wollongong 2522, AUSTRALIA
huaxiong@uow.edu.au

² Department of Computer Science
School of Computing
National University of Singapore
S119260, SINGAPORE
{lamky, xiaogz, zhaohh}@comp.nus.edu.sg

Abstract. As known, multiplicative secret sharing schemes over Abelian groups play an important role in threshold cryptography, such as in threshold RSA signature schemes. In this paper we present a new approach for constructing multiplicative threshold schemes over finite Abelian groups, which generalises a scheme proposed by Blackburn, Burmester, Desmedt and Wild in Eurocrypt'96. Our method is based on a notion of multiple perfect hash families, which we introduce in this paper. We also give several constructions for multiple perfect hash families from resolvable BIBD, difference matrix and error-correcting code.

1 Introduction

A *secret sharing scheme* is a method of protecting a *secret* among a group of *participants* in such a way that only certain specified subsets of the participants (those belonging to the *access structure*) can reconstruct the secret. Secret sharing schemes were first proposed for cryptographic applications, where they can be used as information security primitives for the distribution of trust between a number of entities. In most applications the secret is a highly sensitive piece of data, and the secret sharing scheme is used to control access to this data by requiring certain groups of entities to cooperate in order to retrieve the data. Example applications include controlling access to a bank vault (the participants are bank managerial staff and the secret is the vault combination), installation of high level cryptographic master keys (the participants are senior system managers and the secret is the master key), and enabling a nuclear missile (the participants are presidents and generals, and the secret is the launch code).

A secret sharing scheme is normally initialised by an external trusted *dealer* who securely transfers a piece of information relating to the secret, called a *share*, to each participant in the scheme. A (t, n) -*threshold scheme* is a secret sharing scheme where the access structure consists of all subsets of at least t

(out of a total of n) [6,15]. Secret sharing schemes, and in particular threshold schemes, have become an indispensable basic cryptographic tool in any security environment where active entities are groups rather than individuals [9]. In this paper we consider (t, n) secret sharing schemes over the Abelian groups with an additional property: *multiplicative*. Multiplicative secret sharing schemes over Abelian groups were introduced by Desmedt et al in [11,10], mainly based on the need for designing threshold cryptosystems, as a generalisation of *homomorphic* secret sharing schemes [3]. For example, multiplicative secret sharing schemes over $\mathbf{Z}_{\phi(N)}$, where N is the product of two distinct primes, play a crucial role in RSA threshold decryption and signature schemes (see [11], [10], [5]).

The goal of this paper is to extend the multiplicative secret sharing scheme proposed by Blackburn, Burmester, Desmedt and Wild in Eurocrypt'96 ([5]). Blackburn et al scheme is based on a recursive construction from multiple multiplicative secret sharing schemes, and implicitly applies a perfect hash family (the explicit connections with perfect hash families are due to Kurosawa and Stinson, see [4]). However, their construction requires the execution of independent and multiple rounds of secret sharing schemes with the *same* secret, this might be difficult or impossible in some cryptographic settings, such as MTA-free environment [13]. By introducing multiple perfect hash families, we present a new method of constructing multiplicative secret sharing schemes over finite Abelian groups. Our construction differentiates from Blackburn et al scheme in that the secrets for the multiple-round secret sharing schemes can be different or even independent. We also give several constructions for multiple perfect hash families—mainly combinatorial in nature—from resolvable BIBD, different matrix and error-correcting code.

The paper is organised as follows. In Section 2 we give the basic definition of secret sharing scheme and multiplicative secret sharing scheme. In section 3, we review the previous results. In Section 4, we introduce the notion of multiple perfect hash family and present our new construction. We give several constructions for multiple perfect hash families in Section 5 and conclude the paper in Section 6.

2 Preliminaries

Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a group of n participants, $\mathcal{S}$ denote the set of secrets, and assume the share of P_i is selected from the set $\mathcal{S}_i$. A (t, n) -threshold scheme is a pair of algorithms: the dealer algorithm and the combiner algorithm. For a secret from $\mathcal{S}$, the dealer algorithm applies the mapping

$$\mathcal{D} : \mathcal{S} \rightarrow \mathcal{S}_1 \times \dots \times \mathcal{S}_n$$

to assign shares to participants in $\mathcal{P}$. The combiner algorithm takes the shares of a subset $\mathcal{A} \subseteq \mathcal{P}$ of participants and returns either the secret, if the set $\mathcal{A} \subseteq \mathcal{P}$ and $|\mathcal{A}| \geq t$, or it fails.

$$\mathcal{C} : \bigcup_{P_i \in \mathcal{A}} \{\mathcal{S}_i\} \rightarrow \mathcal{S} \cup \{\text{fail}\}.$$

The classic example of a (t, n) secret sharing scheme is Shamir's scheme [15], which uses Lagrange interpolation on the polynomials over finite fields. It has $\mathcal{S} = \mathcal{S}_i = GF(q)$ and works as follows. To construct a (t, n) threshold scheme ($n < q$) protecting $s \in \mathcal{S}$, choose n distinct non-zero values $x_1, \dots, x_n \in GF(q)$ and define $\mathcal{D}$ as

$$\mathcal{D}(s) = (f(x_1), \dots, f(x_n)),$$

where $f(x_i) = s + r_1x_1 + r_2x_i^2 + \dots + r_{t-1}x_i^{t-1}$ and $r_1, \dots, r_{t-1}$ are $t-1$ random values from $GF(q)$. The values x_i are made public (x_i is associated with P_i). The function $\mathcal{C}$ takes as input at least t valid shares and uses Lagrange interpolation formula to compute $f(x)$ as

$$f(x) = \sum_{i \in B} f(x_i) \prod_{j \in B, j \neq i} \frac{(x - x_j)}{(x_i - x_j)}$$

where $B \subseteq \{1, \dots, n\}$ and $|B| = t$, hence reconstructing the secret $s = f(0)$.

In a multiplicative threshold secret sharing schemes over groups [11], [10], the key space $\mathcal{S}$ is a finite group with respect to the operation $'*'$ and for any t distinct participants $P_i, i \in B$, where $B = \{i_1, i_2, \dots, i_t\} \subseteq \{1, 2, \dots, n\}$, there exists a family $f_{i_1, B}, \dots, f_{i_t, B}$ of functions from $\mathcal{S}_{i_1}, \dots, \mathcal{S}_{i_t}$ to $\mathcal{S}$, respectively, and a public ordering $i_1, \dots, i_t$ of elements of B with the following property. For any key $s \in \mathcal{S}$ and shares $s_{i_1}, \dots, s_{i_t}$ that have been distributed to $P_i, i \in B$ by the dealer algorithm $\mathcal{D}$ on input s , we may express s as:

$$s = f_{i_1, B}(s_{i_1}) * \dots * f_{i_t, B}(s_{i_t}).$$

A multiplicative scheme is *homomorphic* if the share set $\mathcal{S}_i$ has also a group structure and the function $f_{i, B}$ is a group homomorphism from $\mathcal{S}_i$ to $\mathcal{S}$, for all i and B . Shamir's scheme over the finite field $\mathbf{Z}_q$ is a homomorphic scheme, where $f_{i_\ell, B}(s_{i_\ell})$ is defined by

$$f_{i_\ell, B}(s_{i_\ell}) = f(x_{i_\ell}) \prod_{j \in B, j \neq i_\ell} \frac{-x_j}{(x_{i_\ell} - x_j)}.$$

3 Previous Results

To our best knowledge, the first work that explicitly studied multiplicative secret sharing scheme was due to Desmedt and Frankel [11]. The basic idea behind their constructions is to generalise Shamir's (t, n) threshold secret sharing scheme over finite fields to finite Abelian groups and extend Lagrange polynomial interpolation over finite fields to modules over rings. The shortcoming of Desmedt-Frankel (t, n) multiplicative secret sharing scheme is that the size of share for each participant is at least n times of that of the secret, thus it could result in very inefficient scheme for the large group. For details of this approach we refer to [11].

The second approach to multiplicative threshold schemes, due to Blackburn, Burmester, Desmedt and Wild [5] was to recursively execute multiple (t, m)

threshold schemes to build a (t, n) scheme with $n > m$. The explicit connection of perfect hash family and their construction was later observed by Kurosawa and Stinson, and further developed by Blackburn in [4]. Since the goal of this paper is to extend Blackburn et al scheme, in the remainder of this section we briefly review their scheme.

3.1 Blackburn-Burmester-Desmedt-Wild Scheme

We briefly review the definition and basic results on perfect hash family before we describe Blackburn-Burmester-Desmedt-Wild scheme.

A (n, m, w) -perfect hash family is a set of functions $\mathcal{F}$ such that

$$f : \{1, \dots, n\} \longrightarrow \{1, \dots, m\}$$

for each $f \in \mathcal{F}$, and for any $X \subseteq \{1, \dots, n\}$ such that $|X| = w$, there exists at least a function f^X in $\mathcal{F}$ such that f^X is an injection on X , *i.e.* the restriction of f^X on X is one-to-one. For a subset X , if the restriction of a function f on X is one-to-one, then we call f *perfect* on X . We will use the notation $PHF(N; n, m, w)$ for a (n, m, w) perfect hash family with $|\mathcal{F}| = N$. Let $N(n, m, w)$ denote the minimum value N such that a $PHF(N; n, m, w)$ exists. We will be interested in the problem how small can $N(n, m, w)$ be for given n, m and w . In particular, we are interested in the behaviour of $N(n, m, w)$ as a function of n , when m and w are fixed. From [14], we know that for fixed m and w , $N(n, m, w)$ is $\Theta(\log n)$.

Perfect hash family originally arose as part of compiler design—see Mehlhorn [14] for a summary of the early results in this area. They have applications to operating system, language translation system, hypertext, hypermedia, file managers, and information retrieval system—see Czech, Havas and Majewski [7] for the survey of recent results. They have also been applied to cryptography, for example in broadcast encryption [12], in secret sharing [5], threshold cryptography [4, 9].

The idea behind Blackburn et al construction is to combine multiple threshold schemes (for a small number of participants) and a perfect hash family to obtain a new threshold scheme for larger number of participants. That is, if we have an algorithm for a (w, m) threshold scheme and a $PHF(N; n, m, w)$, then we can efficiently construct a (w, n) threshold scheme. The scheme works as follows.

1. On being given a secret $s \in \mathcal{S}$, executes independently the dealer algorithm $\mathcal{D}$ of (w, m) threshold schemes for a secret s with N rounds, where N is the total number of functions in the perfect hash family, and produce elements $c^1, \dots, c^N \in \mathcal{S}_0^m$, where $\mathcal{S}_0$ is the set of shares of the old w out of m threshold. For each $j \in \{1, \dots, N\}$, we write $c^j = (c_{1,j}, \dots, c_{m,j})$, where $c_{jk} \in \mathcal{S}_0$ denotes the k th share of the j th round (w, m) secret sharing scheme.
2. Assign n shares for a (w, n) threshold scheme with participants $\mathcal{P} = \{P_1, \dots, P_n\}$ as follow. We write $\mathcal{F} = \{f_1, \dots, f_N\}$ and the share of P_i is

$$(d_{i,1}, \dots, d_{i,N})$$

defined by

$$d_{i,j} = c_{f_j(i),j}$$

for all $1 \leq i \leq n$ and $1 \leq j \leq N$.

We denote the *old* (w, m) threshold scheme as $(\mathcal{D}_0, \mathcal{C}_0)$ and the *new* (w, n) threshold scheme as $(\mathcal{D}, \mathcal{R})$. Following [5], we assume that the share space of each participants in the old (w, m) threshold scheme are the same, the *share expansion* is then defined to be the rate $\max_{1 \leq i \leq n} \log |\mathcal{S}_i| / \log |\mathcal{S}_0|$. In particular, if the old scheme is ideal, then the share expansion is just a reciprocity of the *information rate* as known in the theory of secret sharing. We thus know that in the above construction the share expansion of the new scheme to the old scheme is N , that is, the number of the functions in the perfect hash family.

Theorem 1. ([5,4]) *Let $(\mathcal{D}_0, \mathcal{C}_0)$ be a perfect (w, m) threshold scheme, and $\mathcal{F}$ is a PHF($N; n, m, w$). Then $(\mathcal{D}, \mathcal{C})$ is a perfect (w, n) threshold scheme and its share expansion (to $(\mathcal{D}_0, \mathcal{C}_0)$) is N . Moreover, $(\mathcal{D}, \mathcal{C})$ is multiplicative provided that $(\mathcal{D}_0, \mathcal{C}_0)$ is multiplicative.*

4 A New Approach

One problem with the Blackburn et al construction is that it requires to execute independent and multiple (w, m) secret sharing schemes with a *same* secret. In some situations (for example, in MTA-free secret sharing [13] it might be difficult or impossible to do so, this leads us to consider the possibility of extending their scheme to cater for this situation.

4.1 Multiple Perfect Hash Families

We first introduce the notion of *multiplicative* perfect hash family which is a straightforward generalisation of perfect hash family. We then extend Blackburn et al schemes based on multiple perfect hash families.

A $k - (n, m, w)$ -*perfect hash family* is a set of functions $\mathcal{F}$ such that

$$f : \{1, \dots, n\} \longrightarrow \{1, \dots, m\}$$

for each $f \in \mathcal{F}$, and for any $X \subseteq \{1, \dots, n\}$ such that $|X| = w$, there exist at least k functions $f_1^X, \dots, f_k^X$ in $\mathcal{F}$ such that f_i^X is an injection on X , *i.e.* the restriction of f_i^X on X is one-to-one, for all $1 \leq i \leq k$. For a subset X , if the restriction of a function f on X is one-to-one, then we call f is *perfect* on X . We will use the notation $k - \text{PHF}(N; n, m, w)$ for a $k - (n, m, w)$ perfect hash family with $|\mathcal{F}| = N$.

4.2 The New Construction

Our new construction is based on multiple perfect hash families and it works as follows.

1. Let $\mathcal{F} = \{f_1, \dots, f_N\}$ be a k -PHF($N; n, m, w$) multiple perfect hash family. We assume that there is a (k, N) threshold scheme $(\mathcal{D}_0, \mathcal{C}_0)$ for a secret set $\mathcal{S}$. For a given secret $s \in \mathcal{S}$, execute the dealer algorithm $\mathcal{D}_0$ of the (k, N) scheme to obtain N shares $a_1, \dots, a_N \in \mathcal{S}_0$ (assuming that all the shares are from a common domain $\mathcal{S}_0$.)
2. For each a_i , $1 \leq i \leq N$, execute the dealer algorithm $\mathcal{D}'$ of a (w, m) scheme $(\mathcal{D}', \mathcal{C}')$ to protect the secret a_i with m shares denoted by $a_{1,i}, \dots, a_{m,i}$.
3. Construct a (w, n) threshold scheme $(\mathcal{D}, \mathcal{C})$ by defining n shares $\mathbf{d}_1, \dots, \mathbf{d}_n$ as

$$\mathbf{d}_j = (a_{f_1(j),1}, \dots, a_{f_N(j),N}),$$

and assign $\mathbf{d}_j$ to participant P_j for all $1 \leq j \leq n$.

Theorem 2. *In the above construction, the resulting (w, n) secret sharing scheme $(\mathcal{D}, \mathcal{C})$ is perfect provided that $(\mathcal{D}_0, \mathcal{C}_0)$ and $(\mathcal{D}', \mathcal{C}')$ are both perfect. Moreover, $(\mathcal{D}, \mathcal{C})$ is multiplicative, provided that $(\mathcal{D}_0, \mathcal{C}_0)$ is homomorphic and $(\mathcal{D}', \mathcal{C}')$ is multiplicative.*

Proof. (Sketch) Assume w participants $P_{i_1}, \dots, P_{i_w}$ want to recover the secret s , using their shares $\mathbf{d}_{i_1}, \dots, \mathbf{d}_{i_w}$. Let $X = \{i_1, \dots, i_w\}$. Since $\mathcal{F}$ is a k -PHF($N; n, m, w$), there exist k perfect hash functions $f_{\ell_1}, \dots, f_{\ell_k} \in \mathcal{F}$ such that f_{ℓ_j} is perfect on $X = \{i_1, \dots, i_k\}$. It follows that $P_{i_1}, \dots, P_{i_w}$ can reconstruct the shares $a_{\ell_1}, \dots, a_{\ell_k}$, using their shares $\mathbf{d}_{i_1}, \dots, \mathbf{d}_{i_w}$. Indeed, applying f_{ℓ_j} to $\mathbf{d}_{i_1}, \dots, \mathbf{d}_{i_w}$, we can retrieve w distinct shares in the secret sharing scheme $(\mathcal{D}', \mathcal{C}')$ associated with the secret a_{ℓ_j} , and so recover a_{ℓ_j} . Since $a_{\ell_1}, \dots, a_{\ell_k}$ consist of k shares of the (k, N) scheme $(\mathcal{D}_0, \mathcal{C}_0)$, and so the secret s can be reconstructed.

Next we show that any $w - 1$ participants have no information about the secret. Indeed, any $w - 1$ participants have at most $w - 1$ shares for each round of (w, m) secret sharing schemes $(\mathcal{D}', \mathcal{C}')$. Since the N rounds of the (w, m) scheme $(\mathcal{D}', \mathcal{C}')$ are independent. It follows that $w - 1$ participants have not information about $a_1, \dots, a_N$, and so have not information about the secret s .

The verification for multiplicative property of $(\mathcal{D}, \mathcal{C})$ is straightforward.

Observe that given a k -PHF($N; n, m, w$), we can composite a (k, N) secret sharing scheme and a (w, m) secret sharing scheme to obtain multiple (w, n) secret sharing schemes. That is

$$(\mathcal{D}_0, \mathcal{C}_0) + (\mathcal{D}', \mathcal{C}') \rightarrow (\mathcal{D}, \mathcal{C}).$$

It is worth noting that Blackburn et al scheme is a special case of the above more general construction. Indeed, if $k = 1$, then $(\mathcal{D}_0, \mathcal{R}_0)$ is just the trivial $(1, N)$ secret sharing scheme, and our construction coincides with theirs.

The efficiency of our general construction is almost the same as Blackburn et al construction, since the complexity of both k -PHF($N_1; n, m, w$) and PHF($N_2; n, m, w$) with respect to N_1 and N_2 as a function of n, m, w are almost the same. However, as we pointed out previously, one advantage of our method is that we do not need to execute multiple secret sharing schemes with a *same* secret.

5 Constructions for Multiple Perfect Hash Families

The resulting (w, n) scheme from the construction in the previous section has information rate Nr_0r' , assuming the information rates for $(\mathcal{D}_0, \mathcal{C}_0)$ and $(\mathcal{D}', \mathcal{C}')$ are r_0 and r' , respectively. Thus the efficiency of our scheme will depend on the size of the perfect hash family N . In this section, we give three explicit constructions (using resolvable BIBD, difference matrix and error-correcting code) for multiple perfect hash families with small sizes of the families.

Let V and U be sets such that $|V| = n$ and $|U| = m$. Recall that a $k - (n, m, w)$ -perfect hash family is a set of functions $\mathcal{F}$ of the form $f : V \rightarrow U$ such that for any $X \subseteq V$ with $|X| = w$, there exist at least k functions $f_1^X, \dots, f_k^X$ in $\mathcal{F}$ such that f_i^X is injection on X , i.e. the restriction of f_i^X on X is one-to-one, for all $1 \leq i \leq k$. Let $k\text{-PHF}(N; n, m, w)$ denote a $k - (n, m, w)$ perfect hash family with $|\mathcal{F}| = N$. we also denote $\text{PHF}(N; n, m, w)$ for $1\text{-PHF}(N; n, m, w)$. From the definition, we know that the perfect hash functions in $\mathcal{F}$ are not necessarily distinct, and thus a k perfect hash family can be trivially constructed from a perfect hash family by repeating k times of each function. Assume that we have a $\text{PHF}(N_0; n, m, w)$, we always have a $k\text{-PHF}(N; n, m, w)$ with $N = kN_0$. The question we are interested in is: can we increase a small value from N_0 , instead of k times of N_0 , to obtain $k\text{-PHF}(N; n, m, w)$? We found several known constructions of perfect hash family have this property.

Before starting our constructions, we make two reformulation for multiple perfect hash family, following [4]. Let $V = \{1, \dots, n\}$. A set $X \subseteq V$ is *separated* by a partition π of V if the elements of X are in distinct parts of π . A $k\text{-PHF}(N; n, m, w)$ is a family $\mathcal{F}$ of N (not necessarily distinct) partitions of V such that each $\pi \in \mathcal{F}$ has at most m parts and such that for all $X \subseteq V$ with $|X| = w$, there are k (also not necessarily distinct) partitions $\pi_1, \dots, \pi_k$ that separate X . Another reformulation is as follows. Let A be an $n \times N$ array, having entries in $U = \{1, \dots, m\}$. We say that the i th column of A *separates* a set X of rows of A if the i th component of the rows in X are all distinct. A $k\text{-PHF}(N; n, m, w)$ is an $n \times N$ array A with entries in the set U which has the property that for every subset X of the row of A with $|X| = w$, there are at least k columns that separate X . As observed in [4]), the three definition for multiple PHF are equivalent.

Our first construction using resolvable Balanced Incomplete Block Design (BIBD) is due to Brickell, we have taken it from [2] and [4]. A (v, b, r, t, λ) -BIBD is an incidence structure, $(V, \mathcal{B})$, having a set V of v points and a family $\mathcal{B}$ of b blocks. Each point is contained in r blocks, each block contains t points and each pair of distinct points is exactly in λ blocks. It follows from elementary counting that $vr = bt$ and $\lambda(v - 1) = r(t - 1)$. A (v, b, r, t, λ) -BIBD, $(V, \mathcal{B})$, is *resolvable* if $\mathcal{B}$ can be partitioned into parallel classes, each of which consists of v/t disjoint blocks.

Theorem 3. Suppose there exists a resolvable (v, b, r, t, λ) -BIBD. Let w be an integer such that $w \geq 2$ and such that $r \geq \lambda \binom{w}{2} + k$. Then there exists a $k\text{-PHF}(\lambda \binom{w}{2} + k; v, v/t, w)$.

Proof. The r parallel classes of the BIBD give rise to r partitions of the point set V of order v , each of these partitions has v/t parts. Let $\mathcal{F}$ be any subset of these partitions of order $r \geq \lambda \binom{w}{2} + k$. Let $X \subseteq V$ be a set of point of order w . Now a partition $\pi \in \mathcal{F}$ fails to separate X if and only if π fails to separate some pair of elements in X . There are at most λ partitions in $\mathcal{F}$ that fail to separate a fixed pair of elements in X . Since there are (exactly) λ parallel classes in the block design that contain the pair of points in a block. Hence there are at most $\lambda \binom{w}{2}$ partitions in $\mathcal{F}$ that fail to separate X . Since $|\mathcal{F}| = \lambda \binom{w}{2} + k$, there are k partitions in $\mathcal{F}$ that separate X . Thus we have showed that $\mathcal{F}$ is a $k - PHF(\lambda \binom{w}{2} + k; v, v/t, w)$.

Corollary 1. ([2]) Suppose q is a prime power and $q \geq \binom{w}{2} + k - 1$. Then there exists a $k - PHF(q + 1; q^2, q, w)$.

Proof. For any prime power q , there exists a resolvable $(q^2, q^2 + q, q + 1, q, 1)$ -BIBD. The result follows from Theorem 3.

The second construction, due to Atici, Magliveras, Stinson and Wei [2], is taken from Blackburn [4]. Recall that a $(q, r; 1)$ -difference matrix is an $r \times q$ matrix $D(d_{i,j})$, having entries in Z/qZ , such that

$$\{d_{i,\ell} - d_{j,\ell} \mid 1 \leq \ell \leq q\} = Z/qZ$$

for all distinct elements $i, j \in \{1, \dots, r\}$. In other words, the vector formed by taking the difference of any two distinct rows of D contains each element of Z/qZ exactly once.

Theorem 4. Suppose there exists a $(q, r; 1)$ -difference matrix $D = (d_{i,j})$, where $r \geq \binom{w}{2} + k$. Then there exists a $k - PHF(\binom{w}{2} + k; q^2, q, w)$.

Proof. Let $V = Z/qZ \times Z/qZ$. Each row $(d_{i,1}, d_{i,2}, \dots, d_{i,q})$ of D gives rise to a partition of V into q parts, $V_1, \dots, V_q$, such that $(a, b) \in V_u$ if and only if

$$d_{i,a} + b = u.$$

Define $\mathcal{F}$ to be any set of $\binom{w}{2} + k$ partitions of V which arise in this way. To prove that $\mathcal{F}$ is a $k - PHF(\binom{w}{2} + k; q^2, q, w)$ it suffices to prove that for any pair of points (a_1, b_1) and (a_2, b_2) in V there is at most one partition arising from D that fails to separate $\{(a_1, b_1), (a_2, b_2)\}$. For suppose that the i th and j th rows of D give rise to partitions of V that fail to separate $\{(a_1, b_1), (a_2, b_2)\}$. Then

$$d_{i,a_1} + b_1 = d_{i,a_2} + b_2 \quad \text{and} \quad d_{j,a_1} + b_1 = d_{j,a_2} + b_2.$$

It follows that $d_{i,a_1} - d_{j,a_1} = (d_{i,a_2} + b_2 - b_1) - (d_{j,a_2} + b_2 - b_1) = d_{i,a_2} - d_{j,a_2}$. Since every element of Z/qZ occurs exactly once in the vector which is the difference of row i and j of D , we have that $a_1 = a_2$. The equality $d_{i,a_1} + b_1 = d_{i,a_2} + b_2$ implies that $b_1 = b_2$. This is a contradiction, since we are assuming that (a_1, b_1) and (a_2, b_2) are distinct.

The third construction is based on error correcting codes. Recall that a (ℓ, M, d, q) -code is a set of M q -ary vectors of length ℓ , with the property that any two vectors in this set differ in at least d positions.

Theorem 5. *Suppose that there exists a (ℓ, M, d, q) -code. Let w be an integer such that $w \geq 2$ and such that*

$$d \geq \frac{((\binom{w}{2}) - 1)\ell + k}{\binom{w}{2}}.$$

Then there exists a $k - PHF(\ell; M, q, w)$.

Proof. Let A denote a $M \times \ell$ array with entries in a q -set such that the rows of A are all the codewords of an (ℓ, M, d, q) -code. We show that A is a $k - PHF(\ell; M, q, w)$ if the conditions of the theorem are satisfied.

Let X be a set of rows of A with $|X| = w$. Since the minimum distance of the code is d , any given pair of distinct rows from X has at most $\ell - d$ columns that the corresponding entries on these two rows are the same. There are $\binom{w}{2}$ possible pairs of distinct rows from X , it follows that there are at least $\ell - \binom{w}{2}(\ell - d)$ columns each of them restrict to the rows of X are all distinct. That is there are $\ell - \binom{w}{2}(\ell - d)$ columns of A that separate X . The condition $d \geq ((\binom{w}{2}) - 1)\ell + k)/\binom{w}{2}$ implies $\ell - \binom{w}{2}(\ell - d)$. The desired result follows.

6 Conclusion

In this paper, we present a new method of constructing multiplicative secret sharing schemes over finite Abelian groups. We introduce the notion of a multiple perfect hash family, which is the basis of our new construction, we also present several constructions of multiple perfect hash families from some combinatorial structures.

References

1. N. Alon and M. Naor, Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, **16** (1996), 434-449.
2. M. Atici, S.S. Magliveras, D. R. Stinson and W.D. Wei, Some Recursive Constructions for Perfect Hash Families. *Journal of Combinatorial Designs* **4**(1996), 353-363.
3. J. C. Benaloh, Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology-Crypto '86*, LNCS, **263** (1986), 251-260.
4. S. R. Blackburn, Combinatorics and Threshold Cryptology. In *Combinatorial Designs and their Applications* (Chapman and Hall/CRC Research Notes in Mathematics), CRC Press, London, 1999, 49-70.
5. S. R. Blackburn, M. Burmester, Y. Desmedt and P. R. Wild, Efficient multiplicative sharing schemes. In *Advances in Cryptology-Eurocrypt '96*, LNCS, **1070** (1996), 107-118.
6. G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of AFIPS 1979 National Computer Conference*, **48** (1979), 313-317.

7. Z. J. Czech, G. Havas and B.S. Majewski, Perfect Hashing. *Theoretical Computer Science*, **182** (1997), 1-143.
8. A. De Santis, Y. Desmedt, Y. Frankel and M. Yung, How to share a function securely, In *Proc. 26th Annual Symp. on the Theory of Computing*, pages 522-533, ACM, 1994.
9. Y. Desmedt, Some recent research aspects of threshold cryptography. In *Information Security Workshop, Japan (JSW '97)*, LNCS, **1396** (1998), 99-114.
10. Y. Desmedt, G. Di Crescenzo and M. Burmester, Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In *Advances in Cryptology-Asiacrypt '94*, LNCS, **917** (1995), 21-32.
11. Y. Desmedt and Y. Frankel, Homomorphic zero-knowledge threshold schemes over any finite group. *SIAM J. Disc. Math.*, **7** (1994), 667-679.
12. A. Fiat and M. Naor, Broadcast encryption, *Advances in Cryptology- Crypto '93*, LNCS, 773:480-490, 1994.
13. W.-A. Jackson, K. Martin and C. M. O'keefe, Mutually trusted authority free secret sharing schemes, *J. Cryptology*, **10**(1997), 261-289.
14. K. Mehlhorn, *Data Structures and Algorithms*. Vol. 1, Springer-Verlag, 1984.
15. A. Shamir, How to Share a Secret. *Communications of the ACM*, **22** (1979), 612-613.
16. D.R. Stinson, An explication of secret sharing schemes. *Des. Codes Cryptogr.*, **2** (1992), 357-390.

On the Security of the RSA-Based Multisignature Scheme for Various Group Structures

Hiroshi Doi¹, Masahiro Mambo², and Eiji Okamoto³

¹ Department of Mathematics, Faculty of Science, Okayama University
hdoi@math.okayama-u.ac.jp

² Education Center for Information Processing & Graduate School of Information
Sciences, Tohoku University
mambo@ecip.tohoku.ac.jp

³ Center for Cryptography, Computer and Network Security,
University of Wisconsin, Milwaukee
okamoto@cs.uwm.edu

Abstract. When more than one user sign a single message, each signer is in a different position in a signing group and has a different responsibility. In such a case, it is important to verify not only the names of participants but also the order of signers. To meet such a requirement a RSA-based multisignature scheme for various group structures was proposed in [5]. This scheme can deal with a group composed of serial and parallel structures. With serial structures different signing orders produce different multisignature and with parallel structures the multisignatures are independent of the signing order. Since the security proofs given in [5] are not complete, it remains open whether the scheme is secure or not. In this paper, we propose a slight modification of the scheme and prove its security by showing reductions to computationally primitive problems.

1 Introduction

A multisignature scheme is a type of digital signature scheme in which more than one user sign a single message. Such a scheme is constructed using a public key cryptosystem like the RSA scheme [9]. Multisignature schemes based on RSA can be divided into two classes. One is a scheme in which all signers use a common modulus as described in [2,3], the other is a scheme in which each signer uses a different modulus as described in [8]. Besides RSA, there is a multisignature scheme [10] based on the ElGamal cryptosystem [6].

Generally speaking, each signer of a multisignature scheme is in a different position in a signing group and has a different responsibility. The signing order often reflects these differences. For example, a signer D supervising the work of a group G signs a document after all members A, B, C of G have signed. By checking the signing order, a verifier knows that D has supervised the creation of the document by G . In such a case, if D signs before other group members, a

generated multisignature should become invalid. On the other hand, the signing order does not have meaning in all cases. It could happen that the signing order in G has no importance. In this case A, B, C are allowed to sign in an arbitrary order and D should sign afterwards. Similar to this example, we can consider concrete examples that involve banks or command structures.

A group is said to be *structured* if the group of signers is structured. The structured group of signers is composed of two structures: 1) *serial structure*, in which the signing order can be detected by a verifier from a signature, 2) *parallel structure*, in which the signing order cannot be detected by a verifier from a signature. By combining these two types of structures, more complicated structure can be constructed.

The signing order of multisignature schemes has not been paid much attention so far except [1,5]. One of schemes in [5] is based on the RSA signature scheme. Hereafter, the scheme is called Doi-Okamoto-Mambo scheme, DOM scheme for short. The security analysis of the DOM scheme is not complete. On the other hand, the scheme in [1] is based on the modified ElGamal and its security is proved by reductions to the discrete logarithm problem.

In this paper we modify the DOM multisignature scheme and give several proofs of security. This paper is organized as follows. After this introduction, we give the notation and an example of the structured group in Sect.2. In Sect.3, we describe the DOM scheme briefly. Then we show that by slightly modifying the scheme as described in Sect.4, we can prove its security as described in Sect.5. The security proof in this context means to give reductions to computationally primitive problems.

2 Notation and Definition

$A, B, C, \dots, U$ are signers. M is a hashed message to sign. Each signer U has his identity, ID , which is composed of two integers. Secret identity (u_1, u_2) expressed by small letter is a pair of secret key of U . In the DOM scheme public identity (U_1, U_2) is expressed by capital letter. Let ISG be information for a structured group. Depending on the context, ISG also denotes the structured group itself. Let $SGID$, structured group identity, be an identity determined from the ISG and the signers' ID s. In contrast to these abbreviations expressed by typewriter face, their instances are expressed by italic face. When we should distinguish symbols of different structured groups, we use a subscript as written by ISG_S , ISG_1 , $SGID_S$, or $SGID_1$. We also use the notation $SGID(ISG_S, (u_1, u_2), \dots)$ for explicitly expressing that $SGID$ is determined from ISG_S and signers' ID s.

For a positive integer k , $\mathbf{Z}_k$ and $\mathbf{Z}_k^*$ are defined as a set $\{i \in \mathbf{Z} | 0 \leq i < k\}$ and a set $\{i \in \mathbf{Z}_k | \gcd(i, k) = 1\}$, respectively. Because all users can treat only a finite range of integers, such a range of integers is denoted by $\mathbf{Z}_{\pm K}$. Furthermore, p, q are primes, and $n = pq$ where p, q, n satisfy conditions required for the RSA scheme. $e \in \mathbf{Z}_{\phi(n)}^*$ and $d \in \mathbf{Z}_{\phi(n)}^*$ are a public exponent and a secret exponent of the RSA, respectively.

Now we describe an outline of the proposed scheme for a structured group G . Since the signing order is taken into account only in the serial structure, the structure of G is restricted to be serial in the following explanation. In the proposed scheme, at first, the first signer A creates A 's signature S_A of M . We call it an intermediate signature of A . Next, using S_A , the second signer B creates S_B of M as an intermediate signature of B . Similarly, each signer after them creates his intermediate signature using the intermediate signature of the previous signer. Finally, an intermediate signature S of the last signer is created, and S is a multisignature of G . We aim to construct a scheme to provide a mechanism of verifying the signing order from S . As shown in the Example 1, there is a close relation between a collusion and a signing order. Therefore we should define the validity of signing order.

Example 1. Let us consider a case such that A, H, B sequentially sign in this order. H is a honest signer and A, B may collude. Then even if A, H, B sign in the reverse order, as long as H performs signature creation for the signing order of A, H, B , a multisignature which passes verification check for A, H, B can be created by the exchange of keys between A and B . Thus, the signing order can be forged.

In Example 1, the faked intermediate signature that a cheated H receives is a right intermediate signature of A in the original signing order A, H, B . Likewise, if more than two colluding signers exchange their keys and arrange their keys in the suitable order, the signing order may be forged. Moreover, the signing order has no meaning if all signers collude, in the similar reason. Based on these observations, we define the validity of signing order.

Definition 1. *Let S be a multisignature of a message M for a structured group ISG_1 . Let H be an honest signer in ISG_1 who obeys the protocol. There exists an algorithm for the verification of S , and S passes the signature verification. S is said to be valid if S satisfies the following three conditions.*

- (V1) *When all the signers in ISG_1 obey the protocol, S is created*
- (V2) *When H does not participate in signature creation of ISG_1 for M , S cannot be created.*
- (V3) *An intermediate signature W which H receives in the process of creating S is exactly the intermediate signature which is created when all the signers obey the protocol. Note that if there exist colluders, they may not obey the protocol.*

From Theorem 4, the probability of collision of SGID is negligible. That means verifying S is substantially a process of checking the signing order. More precisely, from the condition (V3) of Definition 1, one can verify from a valid multisignature at least that if an honest signer exists in the structured group, he has used a right intermediate signature in order to sign the message.

Naturally speaking, before signing a message, each signer should know for which structured group he takes a signing action. Furthermore, checking the name of the previous signer has something to do with the security of the signing

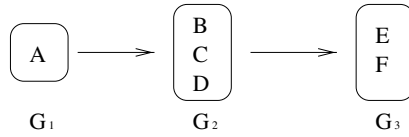


Fig. 1. Example of group structure

order. Indeed, an honest singer may be able to detect the change of the signing order by the identity verification. For instance, the forgery in Example 1 does not succeed if H can find that the previous signer is B and not A .

In this paper, we assume that ISG is given from the previous signer. The previous signer may be an attacker. Furthermore, we assume that the singer cannot check the name of the previous signers. By these assumptions, we can evaluate the security of the proposed scheme in a stronger attack model.

Finally, we give an example of structured group which is used in the whole of this paper.

Example 2. Figure1 shows a structured group composed of serial and parallel structures. In Fig.1, three signing groups G_1, G_2, G_3 are combined. The signing order inside each group is not particularly important. But the different order among three groups yields different multisignature. Information of the structured group shown in Fig.1 is denoted by ISG_G .

3 Doi-Okamoto-Mambo Scheme

We briefly describe the RSA-based multisignature scheme proposed in [5].

3.1 Structured Group ID

Two kinds of basic structured groups, serial and parallel, are considered. General structure is constructed as a combination of these basic structures.

Basic Structures. In the serial structure, a different order of signers causes a different SGID .

Example 3. (Serial structure:) For ISG_1 such that B signs a message after A , $\text{SGID}_1 = ((1 + a_1)a_2 + b_1)b_2$. On the other hand, for a structured group ISG_2 , which has a reversed signing order of ISG_1 , $\text{SGID}_2 = ((1 + b_1)b_2 + a_1)a_2$.

In the parallel structure, the difference of signing order among signers has no importance. Each signer makes his SGID independently and simultaneously.

Example 4. (Parallel structure:) For the simplest structure SGID_P such that each A, B signs the same message in parallel, $\text{SGID}_P = (1 + a_1)a_2 + (1 + b_1)b_2$.

General Structure. We only give an example for the structured group ISG_G shown in Example 2. For the general construction of $SGID$, please refer to [5].

Example 5. $SGID_G$ for ISG_G is generated as follows.

- (DOM-S1) $SGID_A$ computed from A 's secret key is $(1 + a_1)a_2$. $SGID_1$ for G_1 is equal to $SGID_A$.
- (DOM-S2) In G_2 , $SGID_U$ computed from U 's secret key is $(SGID_1 + u_1)u_2$. Thus, $SGID_2$ for G_2 is the sum of these $SGID_U$, i.e. $SGID_B + SGID_C + SGID_D$.
- (DOM-S3) As in (DOM-S2), $SGID_3$ for G_3 is $SGID_E + SGID_F$.

3.2 DOM Protocol

We describe the DOM protocol for a generalized structure ISG_S which contains a group G_i .

Distribution. In the DOM scheme, each user U publishes (U_1, U_2) , and shares U 's first secret key u_1 with a trusted center. U 's second secret key u_2 is generated by a hash function with input of $SGID(ISG_S, (U_{i,1}, U_{i,2}), \dots)$ and u_1 . For each ISG_S , the trusted center generates public keys as follows.

- (DOM-D1) The center creates primes p, q , and publishes $n = pq$. The primes p, q are secret, but there is no need to preserve them after the distribution.
- (DOM-D2) The center creates $SGID_S = SGID(ISG_S, (u_{i,1}, u_{i,2}), \dots)$. Next, the center calculates (v_1, v_2) which satisfies $(SGID_S + v_1)v_2 \equiv 1 \pmod{\phi(n)}$ and publishes (v_1, v_2) .

Signing. Let W be an intermediate signature signed by the users who should sign before U . The signer U in a group G_i signs both the message M and the intermediate signature W using (u_1, u_2) . But W is different depending on U 's position in the structured group. If U is in the first group, he sets $W = M$ because there is no intermediate signature W . If U is not in the first group, he uses W generated by the former group. U 's signature is $(WM^{u_1})^{u_2} \pmod{n}$. When all the signers in G_i finished signing, created signatures are multiplied to generate $\prod_{U \in G_i} (WM^{u_1})^{u_2} \pmod{n}$. This value is an intermediate signature S_{G_i} of G_i . Generated S_{G_i} is passed to the next group. Passing S_{G_i} is a part of the operation for a serial structure, and the multiplication inside G_i is a part of the operation for a parallel structure. S_S generated by the last group is a multisignature in ISG_S .

Verification. The validity of the multisignature S_S is checked by $(S_S M^{v_1})^{v_2} \equiv M \pmod{n}$. If all the signers in ISG_S obey the protocol, $S_S \equiv M^{SGID_S} \pmod{n}$, and S_S is accepted as a multisignature of ISG_S .

Example 6. Now we describe the protocol for ISG_G shown in Example 2.

- Distribution: For ISG_G , the center generates primes p, q and $n = pq$. Then he calculates $SGID_G$ determined from the ISG_G and signer's secret keys. He generates (v_1, v_2) which satisfies $(SGID_G + v_1)v_2 \equiv 1 \pmod{\phi(n)}$ and publishes (n, v_1, v_2) .
- Signing in G_1 : Only one signer A exists in the first group G_1 . Thus, $(MM^{a_1})^{a_2} \pmod{n}$ is A 's signature and also is the intermediate signature S_1 of G_1 .
- Signing in G_2 : In the second group G_2 , there are three signers B, C, D . Each of them receives S_1 and creates their own signature as defined in the protocol. For example, $(S_1 M^{b_1})^{b_2} \pmod{n}$ is B 's signature. After all, by multiplying them, $\prod_{U \in \{B, C, D\}} (S_1 M^{u_1})^{u_2} \pmod{n}$ is generated as the intermediate signature S_2 of G_2 .
- Signing in G_3 : In the last group G_3 , each E, F receives S_2 and creates a signature as defined in the protocol. Then S_3 for G_3 is generated. The multisignature S_G is equal to $\prod_{U \in \{E, F\}} (S_2 M^{u_1})^{u_2} \pmod{n}$.
- Verification: $(S_G M^{v_1})^{v_2} \equiv M \pmod{n}$ is checked. If all the signers obey the protocol, $S_G \equiv M^{SGID_G} \pmod{n}$, and S_G is accepted.

4 Modified Scheme

Apparently, the DOM scheme works for various group structures, but only a few proofs of security are given in [5]. In this section, we propose a modified DOM multisignature scheme. At first, we clarify assumptions for RSA and generation of random number in Sect.4.1. Then, we give strict definitions of $SGID$ and analyze its properties in Sect.4.2. At last, we describe the proposed scheme in Sect.4.3. We give security proofs by using reductions to the problems whose difficulty is assumed in Sect.4.1.

4.1 Assumptions

In this paper, we assume the following conditions for the security of the RSA scheme.

Definition 2. $RSAPhi(e, n)$ is a function that on input $e \in \mathbf{Z}_{\phi(n)}^*$ and $n \in \mathbf{N}_{>1}$, outputs $m \in \mathbf{Z}_{\pm K}$ such that $a^m \equiv 1 \pmod{n}$ for all $a \in \mathbf{Z}_n^*$, if such an m exists.

Definition 3. $RSASig(M, e, n)$ is a function that on input $M \in \mathbf{Z}_n^*, e \in \mathbf{Z}_{\phi(n)}^*$ and $n \in \mathbf{N}_{>1}$, outputs $S \in \mathbf{Z}_n^*$ such that $S^e \equiv M \pmod{n}$, if such an S exists.

Assumption 1. The functions $RSAPhi$ and $RSASig$ are intractable.

Assumption 2. For an arbitrary positive integer k , there exists a random generator whose output is uniformly distributed over $\mathbf{Z}_k$.

Note that m in Definition 2 is a multiple of $\phi(n)$. An attack using $RSAPhi$ is described in pp.94–95 of [7]. We use reductions with respect to expected polynomial-time Turing reducibility, $\leq_T^{ep}$ in our security proof. Polynomial-time version of its definition is described in [1] and omitted here.

4.2 Modification in Structured Group ID

We strictly define a notion of **SGID** computed from a structured group and signers' IDs. By the strict specification of **SGID** we can derive several security proofs as explained in Sect.5. The differences of ID and **SGID** from the DOM scheme is the domain of ID and **SGID**, especially u_2 .

Let **SGID**-space, written as Ω , and ID-space, written as Λ , be a set of **SGIDs** and a set of IDs, respectively. If an ID-space is given, we can define the **SGID**-space as follows.

Definition 4. *If an initial value $s \in Z_{\phi(n)}$ and an ID-space $\Lambda \subseteq \mathbf{Z}_{\phi(n)} \times \mathbf{Z}_{\phi(n)}^*$ are given, **SGID**-space $\Omega \subseteq \mathbf{Z}_{\phi(n)}$ is defined inductively as follows.*

- (i) *For an arbitrary $(u_1, u_2) \in \Lambda$, $(s + u_1)u_2 \pmod{\phi(n)} \in \Omega$.*
- (ii) *For arbitrary $x \in \Omega$ and $(u_1, u_2) \in \Lambda$, $(x + u_1)u_2 \pmod{\phi(n)} \in \Omega$.*
- (iii) *For arbitrary $x, y \in \Omega$, $x + y \pmod{\phi(n)} \in \Omega$.*

$((s_1 + a_1)a_2 + (s_1 + b_1)b_2 + c_1)c_2 \pmod{\phi(n)}$ is an example of **SGID**. If one knows $\phi(n)$, he can calculate **SGID** modulo $\phi(n)$, but only the trusted center knows $\phi(n)$ in our scheme. So, each signer knows only a value in $\mathbf{Z}_{\pm K}$, but not in $\mathbf{Z}_{\phi(n)}$. Now we describe some lemmas used in the later section.

Lemma 1. *If $s = 1$ and $\Lambda = \mathbf{Z}_{\phi(n)} \times \mathbf{Z}_{\phi(n)}^*$, then $\Omega = \mathbf{Z}_{\phi(n)}$.*

Proof. First, we fix some $u_2 \in \mathbf{Z}_{\phi(n)}^*$. For an arbitrary $z \in \mathbf{Z}_{\phi(n)}$, if we set $u_1 = zu_2^{-1} - 1 \pmod{\phi(n)}$, then $z \in \Omega$ by Definition 4(i). So $\mathbf{Z}_{\phi(n)} \subseteq \Omega$. On the other hand, $\Omega \subseteq \mathbf{Z}_{\phi(n)}$ is trivially satisfied. Thus, $\Omega = \mathbf{Z}_{\phi(n)}$ is proved. $\square$

From Lemma 1, if ID-space is maximal, **SGID**-space is maximal too. In this paper, we fix that $s = 1$, ID-space $\Lambda = \mathbf{Z}_{\phi(n)} \times \mathbf{Z}_{\phi(n)}^*$ and **SGID**-space $\Omega = \mathbf{Z}_{\phi(n)}$.

Lemma 2. *When an ISG_S is given, we can construct $SGID_S \in \Omega$ by following the rules in Definition 4. Consider a case such that $(a_1, a_2) \in \Lambda$ is used to construct $SGID_S$. Using $X, Y, Z \in \mathbf{Z}_{\pm K}$, $SGID_S$ is expressed as*

$$(X + a_1)a_2Y + Z, \quad (1)$$

where $X, Y > 0$ and $Z \geq 0$.

Proof. It is easily derived from Definition 4. $\square$

The trusted center can compute $X, Y, Z \pmod{\phi(n)}$. But without the knowledge of $\phi(n)$, all users can deal with only the form of Lemma 2

Apart from the definition of **SGID**, the procedure to compute **SGID** is the same as described in Sect.3.1. Thus, we only point out a problem for a serial structure. In a serial structure, a different order of signers causes a different **SGID**. For almost of all cases in Example 3, $SGID_1 \neq SGID_2$, i.e.

$$((1 + a_1)a_2 + b_1)b_2 \equiv ((1 + b_1)b_2 + a_1)a_2 \pmod{\phi(n)} \quad (2)$$

does not satisfy. Therefore, we can use Definition 4 as a **SGID** of a serial structure. Of course, a probability that (2) satisfies is not equal to 0. However, this probability is very small as shown in Sect.5.3, and it can be ignored.

4.3 New Protocol

We assume the existence of a trusted center which knows all signers' secret keys $(u_1, u_2) \in \Lambda$. The center publishes for each structured group a distinct modulus n of the RSA scheme and a verification key. As in the DOM scheme, all users can create a multisignature after the distribution phase. This distribution occurs only once for a structured group and the same keys can be repeatedly used for the same structured group.

The difference between the proposed modified scheme and the original DOM scheme is only the key distribution, which is influenced from the difference of domain of ID and SGID. Protocols for signing and verification are not different from that in Sect.3.2. Hence, we describe only the key distribution for ISG_S .

Distribution. For a new ISG_S , a center generates keys only once as follows. In contrast to the DOM scheme, public identity of signer is not used in the entire protocol, a hash function is not used for generating signer's secret keys, and the center and U share no secret key in advance.

- (D1) The procedure is the same as in (DOM-D1) explained in Sect.3.2.
- (D2) For each U in ISG_S , the center generates a pair of random integers $(u_1, u_2) \in \Lambda$ as U 's secret key. These random integers u_1, u_2 are generated repeatedly until $u_1 \in \mathbf{Z}_{\phi(n)}$ and $u_2 \in \mathbf{Z}_{\phi(n)}^*$ is satisfied. Once a pair of (u_1, u_2) is generated, the center sends (u_1, u_2) to U in a secure way. Then, the center calculates $SGID_S = SGID(ISG_S, (u_{i,1}, u_{i,2}), \dots)$ as described in Sect.3.1. At last, the center calculates $(v_1, v_2) \in \Lambda$ which satisfies $(SGID_S + v_1)v_2 \equiv 1 \pmod{\phi(n)}$ and publishes them. To calculate (v_1, v_2) , he generates a random integer v_2 satisfying $v_2 \in \mathbf{Z}_{\phi(n)}^*$, and sets $v_1 = v_2^{-1} - SGID_S \pmod{\phi(n)}$.

As shown above, the key published by the center is (n, v_1, v_2) , which differs in each structured group. Note that n must be different in a different structured group in order to avoid attacks known for the RSA signature scheme.

5 Security Considerations

In the proposed scheme, we assume the center is trusted and only users would cheat. Thus, we evaluate the security against a collusion of all the signers except one honest signer H .

The condition (V1) of Definition 1 is guaranteed by the reliable key generation of the trusted center. Concerning the conditions (V2) and (V3) of Definition 1, we examine the security of the modified scheme from the following viewpoints.

Forging multisignature: Colluders try to forge a multisignature without the cooperation of H . This analysis is given in [5] such that forging the RSA scheme reduces to this problem.

Changing the signing order: Colluders pass an incorrect intermediate signature W' to H , and receive H 's signature on W' . With this knowledge colluders try to forge a multisignature.

Attacks to other ISG: With the knowledge of ISG_1 , colluders try to generate a multisignature for ISG_2 .

Collision of SGID: We evaluate the probability that two arbitrary SGIDs of different group structures are congruent to each other.

In this section, modulus is $\phi(n)$ in default and we omit to write it for simplifying the description. From Sect.4.2, we assume $\Lambda = \mathbf{Z}_{\phi(n)} \times \mathbf{Z}_{\phi(n)}^*$ and $\Omega = \mathbf{Z}_{\phi(n)}$. Note that both of secret key (u_1, u_2) and verification key (v_1, v_2) belong to Λ . Generally speaking, $\phi(n) \geq (\ln(2) \cdot n) / \ln(2n)$ is satisfied, and we denote $\rho = \lfloor (\ln(2) \cdot n) / \ln(2n) \rfloor$. Furthermore, we assume that the number of signers is in the polynomial order of $|n|$. Otherwise, the signing process does not terminate in a practical-time scale.

From Lemma 2, colluders without the knowledge of $\phi(n)$ know that

$$((X + h_1)h_2Y + Z + v_1)v_2 \equiv 1 \quad (3)$$

holds for H 's unknown secret key $(h_1, h_2) \in \Lambda$, where $X, Y, Z \in \mathbf{Z}_{\pm K}$ are constant numbers derived from colluders' secret keys, and $(v_1, v_2) \in \Lambda$ is the public key.

We point out that even if $SGID_S$, ISG_S and all the IDs except (h_1, h_2) are given, (h_1, h_2) cannot be determined uniquely.

Lemma 3. *H 's secret key (h_1, h_2) which satisfies (3) cannot be determined uniquely modulo $\phi(n)$. Indeed, there are $\phi(\phi(n))$ candidates of secret keys.*

Proof. Let c be $(X + h_1)h_2$. For arbitrary $x_2 \in \mathbf{Z}_{\phi(n)}^*$, if we set x_1 to $cx_2^{-1} - X$, (x_1, x_2) satisfies (3). This analysis also implies that $x'_1 \in \mathbf{Z}_{\phi(n)} \setminus \{x_2^{-1}c - X \pmod{\phi(n)} \mid x_2 \in \mathbf{Z}_{\phi(n)}^*\}$ cannot be a solution of (3). Because the number of elements in $\mathbf{Z}_{\phi(n)}^*$ is $\phi(\phi(n))$, the lemma is proved. $\square$

We call (x_1, x_2) satisfying (3) a substitution key of (3).

5.1 Changing the Signing Order

In this section, we study an attack to change the signing order. At first, we define a function **ChangeKey** which generates keys for changing the signing order.

Definition 5. *$\text{ChangeKey}(ISG_1, u_{i,1}, u_{i,2}, \dots, \eta, v_1, v_2, n)$ is a function that on input a structured group ISG_1 , colluder's secret keys $(u_{i,1}, u_{i,2}) \in \Lambda$, the number η of substitution keys, a verification key $(v_1, v_2) \in \Lambda$ and a modulus $n \in \mathbf{N}_{>1}$, outputs a structured group ISG_2 and colluders' secret keys $(u'_{i,1}, u'_{i,2}) \in \Lambda$ such that*

$$((X + h_1)h_2Y + Z + v_1)v_2 \equiv 1 \quad , \quad (4)$$

for each of η substitution keys (h'_1, h'_2) of (4),

$$((X' + h'_1)h'_2Y' + Z' + v_1)v_2 \equiv 1, \quad (5)$$

and $X \neq X'$, where (X, Y, Z) and (X', Y', Z') are calculated from $(ISG_1, u_{i,1}, u_{i,2}, \dots)$ and $(ISG_2, u'_{i,1}, u'_{i,2}, \dots)$ as explained in Lemma 2, respectively, and (h_1, h_2) is a secret key of H in ISG_1 , if such outputs exist. Otherwise it outputs $\perp$. Note that (h_1, h_2) may be included in the set of η substitution keys (h'_1, h'_2) of (4).

This function forges a new signing order and/or new secret keys by using H 's signature on an incorrect intermediate signature $M^{X'} \pmod n$. Thus, if **ChangeKey** exists, the validity of S in terms of Definition 1(V3) is not guaranteed.

Note that (4) is derived from published information and all colluders' secret keys. But according to Lemma 3, there are $\phi(\phi(n))$ candidates for H 's secret key satisfying (4). From the existence of such substitution keys, the difficulty of computing **ChangeKey** is classified into several cases:

- A) Computing **ChangeKey** for more than two substitution keys.
- B) Computing **ChangeKey** for one substitution key.

As special cases of A) and B), we can consider the following cases.

- A') Computing **ChangeKey** for all substitution keys.
- B') Computing **ChangeKey** for one substitution key that is exactly the key H actually possesses.

Our concern is whether **ChangeKey** outputs values useful for the successful forgery. In the case A'), whichever key satisfying (4) H actually possesses, the attack using the output of **ChangeKey** is successful with probability 1. Less the number of substitution keys satisfying (5) becomes, smaller the success probability of the attack becomes. In the case B), the success probability of the attack using the output of **ChangeKey** becomes $1/\phi(\phi(n))$, which means computing **ChangeKey** for the case B) is useless in reality. On the other hand, in the case B'), the attack using the output of **ChangeKey** is successful with probability 1. However, additional information is required for narrowing down $\phi(\phi(n))$ candidates of secret keys into one.

We analyze the difficulty of computing **ChangeKey** in the case A) and in the case B'). To this end, **ChangeKeyA** and **ChangeKeyB'** are defined by limiting the input η of **ChangeKey** in accordance with the condition of the case A) and the case B'), respectively. Note that the case A') is included in the case A).

With respect to **ChangeKeyA**, $RSAPhi \leq_T^{ep} \text{ChangeKeyA}$ is proved conditionally in [5]. The condition is that colluders can collect more than three pairs of (X', Y', Z') s for the same $(h'_1, h'_2) = (h_1, h_2)$. We improve this result.

Lemma 4. Suppose that X, Y, Z which satisfy (4) are given. If $X' (\neq X)$, Y' , $Z' \in \mathbf{Z}_{\pm K}$ which satisfy (5) for more than two substitution keys (h'_1, h'_2) of (4) are found, the multiple of $\phi(n)$ can be calculated.

Proof. Each of substitution keys (h'_1, h'_2) of (4) satisfies (4) and (5). Thus, we can derive the congruences

$$h'_1 h'_2 Y v_2 \equiv \bar{Z} - X h'_2 Y v_2 \quad \text{and} \quad h'_1 h'_2 Y' v_2 \equiv \bar{Z}' - X' h'_2 Y' v_2$$

where $\bar{Z} = 1 - Z v_2 - v_1 v_2$ and $\bar{Z}' = 1 - Z' v_2 - v_1 v_2$. Multiply Y' and Y to the former and to the latter, respectively, and delete the term including $h'_1 h'_2$. Then

$$(X' - X) h'_2 v_2 Y Y' \equiv \bar{Z}' Y - \bar{Z} Y'$$

is obtained. Although h'_2 can take more than two arbitrary elements of $\mathbf{Z}_{\phi(n)}^*$, the right hand is a constant. In order to satisfy the identical congruence, $(X' - X) v_2 Y Y'$ must be congruent to 0. Additionally, considering the assumption $X' - X \neq 0$, $Y, v_2 > 0$, we conclude that $(X' - X) v_2 Y Y'$ is some multiple of $\phi(n)$ when $Y' \neq 0$, and $(Z' + v_1) v_2 - 1$ is some multiple of $\phi(n)$ when $Y' = 0$. $\square$

Theorem 1. $\text{RSAPhi} \leq_T^{ep} \text{ChangeKeyA}$.

Proof. Let e and n be inputs of **RSAPhi**. At first, an attacker selects some ISG_1 and generates random integers $u_{i,1}, u_{i,2}, \dots, x \in \mathbf{Z}_\rho$ using a random generator assumed in Assumption 2. η is set to be a suitable number which is equal to or greater than two. Until **ChangeKeyA** outputs an answer other than $\perp$, he calculates **ChangeKeyA**($ISG_1, (u_{i,1}, u_{i,2}), \dots, \eta, x, e, n$) repeatedly.

Let α be the number of participants of the group. From

$$\frac{\phi(\rho)}{\rho} \geq \frac{\ln(2) \cdot \rho}{\ln(2\rho) \cdot \rho} = \frac{\ln(2)}{\ln(2\rho)},$$

the attacker expects to obtain $u_{i,2}$ which satisfies $u_{i,2} \in \mathbf{Z}_{\phi(n)}^*$ for α signers if he tries $(\ln(2\rho)/\ln(2))^\alpha$ times. Because ρ is $O(n/\ln(n))$, the trial time $(\ln(2\rho)/\ln(2))^\alpha$ is $O((\ln(n) - \ln(\ln(n)))^\alpha)$ and the order is bounded above by a polynomial of $|n|$.

By using inputs of **ChangeKeyA**, the attacker can calculate (X, Y, Z) satisfying (4), and by using the answer of **ChangeKeyA**, he can calculate (X', Y', Z') satisfying (5) for at least two substitution keys of (4). Therefore, the multiple of $\phi(n)$ can be calculated from Lemma 4. $\square$

Regarding the case B'), we examine the complexity of **ChangeKeyB'** in the situation such that an attacker can additionally have an access to an RSA signing oracle which uses the same modulus as in the proposed scheme under the attack. We define a function **ChangeKeyB'Info** for computing **ChangeKeyB'** with information on multisignature.

Definition 6. **ChangeKeyB'Info**($ISG_1,$

$u_{i,1}, u_{i,2}, \dots, 1, v_1, v_2, n, (M_i, X_i, S_i),$

$\dots$) is a function which behaves the same as **ChangeKey** except that (5) in Definition 5 holds only for the substitution key satisfying $(h'_1, h'_2) = (h_1, h_2)$, and that it has additional inputs of a polynomial number of multisignatures (M_i, X_i, S_i) . Note that from the first condition for exception, η is set to be 1.

Theorem 2. *In the situation such that an attacker can have an access to a RSA signing oracle which uses the modulus of the target multisignature scheme, we can prove $\text{RSAPhi} \leq_T^{ep} \text{ChangeKeyB'Info}$.*

Proof. Let e and n be inputs of RSAPhi . At first, an attacker selects some ISG_1 , and generates random integers $u_{i,1}, u_{i,2}, \dots, x \in \mathbf{Z}_\rho$. Then he can calculate X, Y, Z of (4). Until satisfying $\gcd(Y, e) = 1$, he repeatedly generates $u_{i,1}, u_{i,2}, \dots, x$. If $\gcd(Y, e) = 1$, he can compute positive integers k and x which satisfy $ek - xY = 1$, using the extended Euclidean algorithm. Next, he sets $(h_1, h_2) = (-X - d, x)$ and $(v_1, v_2) = (k - Z, e)$. It is easy to check that (h_1, h_2) , X, Y, Z, v_1 and v_2 satisfy (4) since $ed \equiv 1$. Thus, $(-X - d, x)$ is regarded as the secret key. By accessing to the RSA oracle with a secret key d corresponding to e and a public modulus n , the attacker obtains a polynomial number of M_i^d for arbitrary M_i . Then he can prepare a polynomial number of (M_i, X_i, S_i) by computing $S_i \equiv (X_i M_i^{-X-d})^x \equiv (X_i M_i^{-X} (M_i^d)^{-1})^x \pmod{n}$ for M_i and arbitrary X_i . Hence, the attacker repeatedly generates $u_{i,1}, u_{i,2}, \dots, x \in \mathbf{Z}_\rho$, which satisfies $\gcd(Y, e) = 1$, and a polynomial number of M_i^d until $\text{ChangeKeyB'Info}(ISG_1, u_{i,1}, u_{i,2}, \dots, v_1, v_2, n)$ outputs an answer other than $\perp$. From the outputs of ChangeKeyB'Info , X', Y', Z' is calculated easily. Using $X', Y', Z', h_1, h_2, v_1$ and v_2 , (5) can be expressed as $((X' + (-X - d))xY' + Z' + k - Z)e \equiv 1$. That is,

$$((X' - X)xY' + Z' - Z + k)e - xY' - 1 \equiv 0. \quad (6)$$

Note that if the left hand is not equal to 0, the attacker can get the multiple of $\phi(n)$ other than 0. If the left hand is equal to 0, the attacker iterates this procedure until the left hand is not equal to 0.

At the end of this proof, we evaluate the number of trials of the above procedure. If we suppose that $X', Y', Z' \in \mathbf{Z}_{\pm K}$ are random outputs, probabilities such that $X' > X, Y' > 0$, and $Z' > Z + k$ are $1/4, 1/2$, and $1/4$, respectively. Indeed, if we suppose that X' is a random integer in $\mathbf{Z}_{\pm K}$ and X is a positive random integer in $\mathbf{Z}_{\pm K}$, we can evaluate that $\text{Prob}(|X'| > |X|)$ is $1/2$, and $\text{Prob}(X' > 0)$ is $1/2$. So $\text{Prob}(X' > X) = 1/4$. Other probability can be calculated in the same manner. If $X' > X, Y' > 0$ and $Z' > Z + k$ are satisfied, the left hand of (6) becomes positive. That means we can expect to obtain X', Y', Z' , by which the left hand of (6) becomes positive, after 32 trials. Therefore, $\text{RSAPhi} \leq_T^{ep} \text{ChangeKeyB'Info}$. $\square$

Let $\text{Ord}_\tau(g)$ denote the multiplicative order of g modulo τ . $\Gamma(\tau) = \{g_{\max} \mid \text{Ord}_\tau(g_{\max}) = \max_g \text{Ord}_\tau(g), \text{ where } g, g_{\max} \in \mathbf{Z}_\tau\}$. $g_{\max, \tau} \in \Gamma(\tau)$. Now, we slightly generalize ChangeKey as follows.

Definition 7. $\text{ChangeFunc}(M, ISG_1, u_{i,1}, u_{i,2}, \dots, \eta, v_1, v_2, n)$ is a function that on input a message $M \in \mathbf{Z}_{\phi(n)}^*$, a structured group ISG_1 , colluder's secret keys $(u_{i,1}, u_{i,2}) \in \Lambda$, the number η of substitution keys, a verification key $(v_1, v_2) \in \Lambda$, and a modulus $n \in \mathbf{N}_{>1}$, outputs an intermediate signature $W' \in \mathbf{Z}_{\phi(n)}^*$ and a function F such that $(F((W' M^{h'_1})^{h'_2}) M^{v_1})^{v_2} \equiv M \pmod{n}$ for $W' \not\equiv M^X \pmod{n}$ and each of η substitution key (h'_1, h'_2) of (4), where $ISG_1, (u_{i,1}, u_{i,2}), (v_1, v_2)$, and n satisfy (4), n additionally satisfies $|\Gamma(n)|/n = 0$

($\text{poly}(|n|)$), X is calculated from $ISG_1, u_{i,1}, u_{i,2}, \dots$ as explained in Lemma 2, if such W' and F exist. Otherwise it outputs $\perp$. Note that (h_1, h_2) may be included in the set of η substitution keys (h'_1, h'_2) of (4).

After receiving W' and F from **ChangeFunc**, an attacker sends W' to an honest signer H and receives H 's signature on M and W' . Then the attacker can forge a multisignature using H 's signature and the function F .

We give the proof of the security for the attack of **ChangeFunc** only in the case A'). The corresponding function is denoted by **ChangeFuncA'**. The security against the attack of **ChangeFunc** in the case of A) other than A') and in the case of B') is left as an open problem.

Definition 8. RSASig' is RSASig with an additional condition such that the input n satisfies $|\Gamma(n)|/n = O(\text{poly}(|n|))$.

Theorem 3. $\text{RSASig}' \leq_T^{ep} \text{ChangeFuncA}'$.

Proof. Let M, e and n be inputs of RSASig' . At first, an attacker sets $v_2 = e$. As in the proof of Theorem 1, he selects ISG_1 and $(u_{i,1}, u_{i,2}), \dots, x \in \mathbf{Z}_\rho$ until **ChangeFuncA'**($M, ISG_1, u_{i,1}, u_{i,2}, \dots, \phi(\phi(n)), x, e, n$) outputs an answer other than $\perp$. Note that from the condition of A'), η is set to be $\phi(\phi(n))$. The expected number of the trials is bounded by a polynomial in $|n|$. Then the attacker obtains W' and F . By giving W' to the target signer H , he can receive an output $(W'M^{h_1})^{h_2} \pmod n$ from H , where (h_1, h_2) is one of substitution keys mentioned in Lemma 3. As the definition of **ChangeFuncA'**, F can compute for all substitution keys a multisignature which passes the verification check. Hence, the attacker can simulate the H 's signing operation by selecting an intermediate signature W'' of H from $\mathbf{Z}_n$ uniformly at random. As long as W'' belongs to a set of H 's outputs, F computes a multisignature S which passes the verification check, i.e. $S = F(W'')$. Then $SM^x \pmod n$ is the output of RSASig' .

Now, we give a very rough estimate for the probability that selected W'' belongs to the set of H 's outputs. Because the substitution key (h'_1, h'_2) satisfies $(X + h'_1)h'_2 \equiv (X + h_1)h_2$,

$$(W'M^{h'_1})^{h'_2} \equiv (W'M^{-X})^{h'_2} M^{(X+h_1)h_2} \pmod n$$

is derived. Because h'_2 is an arbitrary integer in $\mathbf{Z}_{\phi(n)}^*$, if $\text{Ord}_n(W'M^{-X})$ is large, the candidates of $(W'M^{h'_1})^{h'_2}$ increase. Since the adequate modulus n satisfies $|\Gamma(n)|/n = O(\text{poly}(|n|))$, $\text{Prob}(W'' \in \Gamma(n)) = O(\text{poly}(|n|))$ and $\text{Prob}(\text{Ord}_n(W'M^{-X}) = \text{Ord}_n(g_{\max, n})) = O(\text{poly}(|n|))$ under the assumption that W' is random integer in $\mathbf{Z}_{\phi(n)}^*$. That means both the trials is bounded by a polynomial in $|n|$ and the probability that F outputs S for a random integer W'' is bounded by a polynomial in $|n|$. Therefore, $\text{RSASig}' \leq_T^{ep} \text{ChangeFuncA}'$. $\square$

5.2 Attacks to Other ISG

In this section, we consider the attacks to ISG_2 using all information related to other ISG_1 . The modulus of ISG_1 and ISG_2 are denoted by n_1 and n_2 , respectively. If all the signers in ISG_1 collude, $\phi(n_1)$ can be easily calculated and may be used to attack against ISG_2 . But the modulus in ISG_2 is different from n_1 . It is not known if there is an attack for the different modulus in the RSA scheme. Thus, we consider it is difficult to perform such an attack.

In the same argument, we assume that an attacker cannot generate a multi-signature for ISG_2 by converting S_H created by an honest signer H in ISG_1 .

5.3 Collision of SGID

Collision of **SGID** for distinct structured groups causes a failure of verifying the signer's group. There are collisions of **SGID** in the proposed scheme. For example, the probability that (2) holds is not equal to 0.

Theorem 4. *Let $SGID_1$ be the **SGID** determined by ISG_1 and $(u_{1,1}, u_{1,2}), \dots$, which are selected uniformly at random from Λ , and let c be an arbitrary constant. Then $\text{Prob}(SGID_1 \equiv c \pmod{\phi(n)}) = 1/\phi(n)$.*

Because n is the modulus of RSA, $\phi(n)$ is sufficiently large and the probability in Theorem 4 can be regarded as negligible.

It is desirable that all the intermediate **SGID** is not equal. In such a case, we must consider the birthday paradox and design the size of n . The number of signers whose **SGIDs** are all distinct is bounded by a polynomial in $|n|$. The same property is observed in [1].

5.4 Problems of Variations

To modify the proposed scheme must be done with greatest care. For example, as explained in [5], the direct application of the threshold technique in [3] to the proposed scheme is insecure because signers involving in the threshold scheme can compute a multiple of $\phi(n)$, where n is a modulus of the multisignature scheme.

Moreover, the following simplification of secret keys should be avoided. In our scheme, the secret key of U is a pair of random integers $(u_{i,1}, u_{i,2})$. If we set the former parameter to be a constant, for example $u_{i,1} = 1$ as discussed in [4], the signing order can be manipulated in the sense of Definition 1(V3) as follows.

Example 7. Consider a structured group in which A, H, B sign in this order. Now suppose that secret key of A, H and B are $(1, a_2)$, $(1, h_2)$ and $(1, b_2)$, respectively. If $2a_2 + 1 = 3k$ satisfies for some integer k , colluding A and B can succeeds in the following attack.

- (F1) An attacker passes $M^2 \pmod n$ as A 's signature S_A to H .
- (F2) H generates his signature $S_H \equiv (S_A M)^{h_2} \pmod n$.
- (F3) An attacker calculates $S_B \equiv (S_H^k M)^{b_2} \pmod n$.

Because $S_B \equiv M^{(((1+1)a_2+1)h_2+1)b_2} \pmod n$, S_B is accepted. We can extend the attack by use of expressions other than $2a_2+1=3k$. In the proposed scheme, such an attack cannot be successful because of Theorem 1.

6 Conclusion

In this paper, we have improved the RSA-based multisignature scheme proposed in [5] and proved its security. We gave the strict definition of the **SGID** determined from the structured group and **ID**. Using **SGID**, we have constructed a multisignature scheme which is suitable for the various group structures composed of serial and parallel. In the proposed scheme one can verify the signing order in a structured group. For the security analysis we have adopted an attack model such that all the signer except one honest signer collude. Based on such an attack model, we have clarified many security properties of the proposed scheme in addition to the proof shown in [5]. We have given security proofs for the attacks of changing the signing order. There are two models for the attacks. One is to generate keys for changing the signing order, and other is to generate functions for changing the signing order. The problem of finding a multiple of $\phi(n)$ of the RSA scheme reduces to the problem of achieving the former. Meanwhile, the same problem can reduce to the problem of achieving the later only in limited situations. The reduction to the later model in the special case is left as an open problem. We have also evaluated the security against the attacks to other **ISG**, and the collision of the **SGID**. Furthermore, the troubles occurred by simplifying key and by directly applying threshold scheme to the proposed scheme have been explained briefly.

References

1. Burmester, M., Desmedt, Y., Doi, H., Mambo, M., Okamoto, E., Tada, M., and Yoshifuji, Y.: A Structured ElGamal-Type Multisignature Scheme, LNCS 1751, Springer-Verlag, Public Key Cryptography, pp. 466–482 (2000).
2. Boyd, C.: Some Applications of Multiple Key Ciphers, LNCS 330, Springer-Verlag, Advances in Cryptology - Eurocrypt'88, pp. 455–467 (1988).
3. Desmedt, Y. and Frankel, Y.: Shared generation of authenticators and signatures, LNCS 576, Springer-Verlag, Advances in Cryptology - Crypto '91, pp. 457–469 (1991).
4. Doi, H., Okamoto, E., Mambo, M. and Uyematsu, T.: Multisignature Scheme with Specified Order, SCIS94-2A, January 27-29, 1994.
5. Doi, H., Okamoto, E. and Mambo, M.: Multisignature Schemes for Various Group Structures, *Proc. of 36th Annual Allerton Conference on Communication, Control and Computing*, pp. 713–722 (1999).

6. ElGamal, T.: 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).
7. Koblitz, N.: *A Course in Number Theory and Cryptography*, GTM 114, Springer-Verlag, 2nd edition (1994).
8. Okamoto, T.: A Digital Multisignature Scheme Using Bijective Public-Key Cryptosystems, *ACM Trans. on Computer Systems*, Vol. 6, No. 8, pp. 432-441 (1988).
9. Rivest, R. L., Shamir, A. and Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems, *Commun. of the ACM*, Vol. 21, No. 2, pp. 120-126 (1978).
10. Yen, S. and Lai, C.: New Digital Signature Scheme Based on Discrete Logarithm, *Electronics Letters*, Vol. 29, No. 12, pp. 1120-1121 (1993).

Fail-Stop Confirmer Signatures

Yi Mu and Vijay Varadharajan

School of Computing and IT,
University of Western Sydney, Nepean, Kingswood, NSW 2747, Australia
{yimu, vijay}@cit.nepean.uws.edu.au

Abstract. A designated confirmer signature allows the signer to prove to the signature's recipient that the designated confirmer can confirm the signature without the signer. In this paper we propose a fail-stop confirmer signature scheme based on the concept of fail-stop knowledge proofs and signature knowledge proofs on discrete logarithms. We also develop a blinded version of the confirmer scheme. The new confirmer signatures have enhanced security against forgery from powerful adversaries.

1 Introduction

An undeniable signature requires the signer to verify the signature for the signature recipient[1]. Designated confirmer signatures[2,3,4] was first proposed by Chaum[2] to allow a undeniable signature to have more flexibility by allowing a designated confirmer to verify the signature for the signer. Actually, Designated confirmer signatures can be designed for both signer and confirmer to verify a signature[5].

Fail-stop signature scheme[6,7,8,9] was proposed by Heyst and Petersen[6] to enhance the security against the adversaries who have a high degree of computational power. Under such scheme, if the secret key of a system is compromised, a designated trusted authority can find the forger related to the instance with an overwhelm probability. The fail-stop mechanism is based on a special public-secret key pair. In particular, the unique public key is designed in such a way that it maps q different solutions on the secret key under a finite field of size q . Since only one solution is registered with the authority. The probability for a powerful adversary to find the correct key is $1/q$.

In this paper, we propose a new confirmer signature scheme that combines the confirmer signatures with a fail-stop method. In the confirmer signature scheme, verification and confirmation of signatures require using techniques of zero-knowledge proofs on discrete logarithms. To make a confirmer signature fail-stop, we need to develop technique that allows the zero-knowledge proofs to fail-stop. That is, if an adversary forged a proof, he or she can then be caught after the instance. We also give the blind version for our fail-stop confirmer signature scheme. These schemes have potential applications in electronic commerce.

The rest of the paper is organized as follows: Chapter 2 introduces our model including the definitions of basic protocols to be used in the basic protocols.

Chapter 3 presents the concepts and protocols of fail-stop knowledge proofs and fail-stop signature knowledge. Chapter 4 gives the detailed protocol of fail-stop confirmer signatures and its blind version. Chapter 5 concludes the paper.

2 Model

We now look at a general construction of our fail-stop confirmer signatures, which is based on the confirmer signature scheme[2] and fail-stop signature scheme[6]. For simplicity, we assume that there exist a signer S , a confirmer C , a signature verifier V , and a trusted third party TTP who helps signers and confirmers to establish their keys.

In our system, both signature and confirmation have the fail-stop property associated to the concepts of knowledge proofs and signature knowledge proofs on discrete logarithms[10,11]. In the following, we give the definitions of “fail-stop” related to those concepts.

Definition 1. (*Fail-Stop Knowledge Proof of Discrete Logarithms*) Given a legitimate knowledge proof $\mathcal{P}$ on a secret associated with a finite field $\mathbb{Z}_q$ and the corresponding verification algorithm $\mathcal{V}$, a powerful adversary can generate a forged proof $\mathcal{P}'$ that can also be verified using the same $\mathcal{V}$. The probability of finding the forgery from the adversary is $1 - 1/q$ (See the next section for proof).

Definition 1 stems from the original fail-stop concept but used to knowledge proofs. We also use the concept of signature knowledge, which is equivalent to digital signatures[11].

Definition 2. (*Fail-Stop Signature Knowledge of Discrete Logarithms*) Given a legitimate signature knowledge $\mathcal{S}$ with the secret key associated to a finite field $\mathbb{Z}_q$ and the corresponding verification algorithm $\mathcal{V}$, a powerful adversary can generate a forged signature knowledge $\mathcal{S}'$ that can also be verified using the same $\mathcal{V}$. The probability of finding the forgery from the adversary is $1 - 1/q$ (see the next section for proof).

The scheme consists of the following protocols.

- There are two key generators. For an input 1^ℓ with a security parameter ℓ ,
 - $FG_S(1^\ell) \rightarrow (K_S, P_S)$, the fail-stop secret-public key pair for the signer. K_S and P_S are respectively a set containing related data.
 - $FG_C(1^\ell) \rightarrow (K_C, P_C)$, the fail-stop secret-public key pair for the confirmer. K_C and P_C are a set respectively.
- There is a probabilistic polynomial knowledge proof algorithm that receives a secret and outputs a fail-stop knowledge proof κ .
- There is a non-interactive verification protocol, $(\mathcal{P}_C, \mathcal{V}_V)$, for signature knowledge proof, between the confirmer and the verifier

$$[\mathcal{P}_C(K_C), \mathcal{V}_V()](\kappa, P_C) \rightarrow \delta, \text{ for } \delta \in \{0, 1\}.$$

$\mathcal{P}_C$ is the proof protocol and $\mathcal{V}_V$ is the corresponding verification protocol. The private input of $\mathcal{P}_C$ is K_C that is the secret to be proved. The common inputs for the protocol are κ, P_C . P_C is the public key for the prover. The verification protocol returns a Boolean value δ . If $\delta = 1$, the κ is valid, and 0 otherwise.

- There is a probabilistic polynomial signature knowledge algorithm, which receives a secret key and a message, outputs a fail-stop signature knowledge σ . There is also a corresponding blind algorithm, $\hat{\sigma}$.
- There is a non-interactive verification protocol, $(\mathcal{S}_S, \mathcal{V}_V)$, for signature knowledge proof, between the signer and the verifier

$$[\mathcal{S}_S(K_S), \mathcal{V}_V()](m, \sigma, P_S, y_c) \rightarrow \delta, \text{ for } \delta \in \{0, 1\}.$$

$\mathcal{S}_S$ is the signing protocol and $\mathcal{V}_V$ is the corresponding verification protocol. The private input of $\mathcal{S}_S$ is K_S . The common inputs for the protocol are m, σ, P_S, y_c . P_S is the signer's public key. y_c is the confirmer's public key. The verification protocol returns a Boolean value δ . If $\delta = 1$, the σ is valid, and 0 otherwise. There is an interactive verification protocol for the blind version defined in a similar manner,

$$[\hat{\mathcal{S}}_S(K_S), \mathcal{V}_V()](m, \hat{\sigma}, P_S, y_c) \rightarrow \delta, \text{ for } \delta \in \{0, 1\}.$$

We have used $\hat{\cdot}$ to denote “blind”.

- There is a protocol between the signer and the *TTP* for finding a possible forger,

$$[\mathcal{T}()](\Sigma_i, P_i) \rightarrow \delta, \text{ for } \delta \in \{0, 1\}.$$

$i = S$ for the signer and $i = C$ for the confirmer. Σ_S denotes σ and Σ_C denotes κ . The common inputs for the protocol are m, Σ_i . If it outputs 1, Σ_i is valid, and 0 otherwise.

Throughout this paper, we will use the following notations:

- $p = 2q + 1$ is a large prime such that q is a prime as well. both p and q is known to the public.
- g is a primitive selected from $\mathcal{Z}_p^*$, a multiplicative group of order q .
- There exists a public one-way function $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ ($l \approx 160$).
- $|$ denotes a modulo operator.

3 Fail-Stop Knowledge Proofs and Signature Knowledge

Knowledge proofs of discrete log were initially introduced by Chaum[10]. Since then, there have been many works on such knowledge proofs. Basically, there are two types of discrete log knowledge proofs: interactive and non-interactive. They have equivalent functionality. Recently, discrete log knowledge proofs have been applied to signature knowledge proofs, which are proved equivalent to digital signatures [11].

An important step in achieving fail-stop confirmer signatures is to make discrete log knowledge proofs *fail-stop*. We will use the fail-stop signature method developed by Chaum to our discrete log proofs. The original purpose of fail-stop signatures is to prevent any powerful adversary from computing the secret key for signing. Under a fail-stop signature, the legitimate signing key is chosen for a large set where all elements map to the public key. Therefore, the powerful opponent has a very small probability of finding the real signing key and a high probability to get a fake key. If one of fake keys is used to sign, then the illegal signer will be caught by the *TTP*.

3.1 Fail-Stop Knowledge Proofs on Discrete Logarithms

As defined in the proceeding chapter, fail-stop knowledge proofs have the same mechanism as that of a fail-stop signature. We will restrict ourselves to non-interactive knowledge proofs only, since they add less load to the network traffic when compared to the interactive methods.

Let $p = 2q + 1$ be a prime such that q is prime as well. Let $g \in \mathcal{Z}_p^*$ be an element of order q . Select α from $\mathcal{Z}_q$ and define $h = g^\alpha|p$. The value of p, q, g, h , and α are chosen by the *TTP*. p, q, g , and h are public and the value of α is secret from others.

We require the prover to have a certificate. The prover must use his certificate during proof construction. The certificate is defined as $Cert \leftarrow (b_1, b_2)$ signed by the *TTP*, where b_1 and b_2 must have a form that suits the fail-stop computation. In order to construct suitable b_1 and b_2 , the prover chooses $x_1, \dots, x_4$ from $\mathcal{Z}_q$ as her secret key at random and computes $b_1 \leftarrow g^{x_1} h^{x_4}|p$ and $b_2 \leftarrow g^{x_2} h^{x_3}|p$.

Let x be the secret to be proved by the prover and $y = g^x|p$ be her commitment to the secret. Note that proving the discrete log, $\log_g y$, is equivalent to proving $\log_{b_1} b_1^x$, provided that we can prove that these two discrete logs are equal. Proving equality of two or more discrete logs is easy by using Fujioka and Okamoto's interactive bi-proof[13]. The bi-proof method can readily be converted to a non-interactive version. For simplicity, we leave the detail to the reader. We always assume the base for the proof is b_1 . That is, $y = b_1^x|p$. The key (including the secret to be proved) for the prover is defined as $K \leftarrow (K_P, P_P)$ for $K_P \leftarrow (x, x_1, \dots, x_4)$ and $P_P \leftarrow (y, b_1, b_2)$.

The protocol to prove knowledge on x from y is as follows:

– Prover:

- Selects¹ $r_1, r_2 \in_R \mathcal{Z}_q$ and computes $d = g^{r_1} h^{r_2}|p$ as a proof commitment.

¹ The original Heyst-Pedersen fail-stop signature scheme uses part of secret key (x_2, x_3) as the signature commitment, We should point out it can be broken using a well-known common signature commitment attack. For example, if we sign two different messages, m_1 and m_2 , the fail-stop signatures are $a_1 = x_2 + m_1 x_1$, $a_2 = x_3 + m_1 x_4$ and $a'_1 = x_2 + m_2 x_1$, $a'_2 = x_3 + m_2 x_4$, the secret key can then be computed with $x_1 = (a_1 - a'_1)/(m_1 - m_2)$ and $x_4 = (a_2 - a'_2)/(m_1 - m_2)$. Adding r to the fail-stop signature can easily fix the problem. This kind of attacks can be applied to all the similar signature schemes such as ElGamal signatures, Schnorr signatures, etc.

- Calculates $c = H(g, h, y, b_1, b_2, d)$.
- Then calculates $a_1 = r_1 + x_2 + x_1 c x$ and $a_2 = r_2 + x_3 + x_4 c x$ for $A \leftarrow (a_1, a_2) \in \mathcal{Z}_q \times \mathcal{Z}_q$.
- Verifier:
 - Checks correctness of the hash function.
 - Checks $dy^c b_2 \stackrel{?}{=} g^{a_1} h^{a_2} | p$.

The completeness of the protocols is obvious, since it is straightforward to see that the verifier can verify the correctness of the proof if the proof is correctly constructed. We now explain the soundness of the protocol in the following lemmas. First of all, we show that the prover has to keep her proof commitment:

Lemma 1. *There exists no polynomial time algorithm, which, on input (d, c, a_1, a_2) , outputs (d', c', a'_1, a'_2) such that $d' y^{c'} b_2 = g^{a'_1} h^{a'_2} | p$.*

Proof. (Sketch) This is because the proof commitment has been embedded in the hash function that forms c . It is easy to see that if d is not in the hash function, the lemma will become incorrect. This is because by giving $d' = y^{-c'} b_2^{-1} g^{a'_1} h^{a'_2} | p$, we obtain $d' y^{c'} b_2 = g^{a'_1} h^{a'_2} | p$ for arbitrary a'_1, a'_2 .

The proof commitment can be used for the specific proof only, otherwise the prover's secret key will be compromised (see the footnote).

Suppose that the powerful opponent (say, Eve) can find $x'_1, \dots, x'_4$ such that $b_1 \leftarrow g^{x'_1} h^{x'_4} | p$ and $b_2 \leftarrow g^{x'_2} h^{x'_3} | p$. The opponent can of course use the same certificate to illegally prove any discrete log at will and her proof can still be verified.

Lemma 2. *Let tuple (x'_1, x'_2, x'_3, x'_4) satisfying $b_1 = g^{x'_1} h^{x'_4}$ and $b_2 = g^{x'_2} h^{x'_3}$ be a fake secret key for the opponent. Then the opponent can prove her knowledge on any x' . However, with probability $1 - 1/q$, the opponent can be caught.*

Proof. (Sketch) We know that there are q discrete log solutions with respect to x_1, x_2, x_3, x_4 but only one registered with the *TTP*[12]. Therefore, the probability finding the correct one by Eve is $1/q$. Let us now see how Eve is caught. To do this, the *TTP* needs to find out α that is known only to the *TTP*. We assume that the opponent produced a proof with a tuple $(x', c', d', A', y', b_1, b_2)$ such that $[\mathcal{P}_{P'}(K_{P'}), \mathcal{V}_V()](\kappa', P_{P'}) \rightarrow 1$ or $A' y'^{c'} b_2 = g^{a'_1} h^{a'_2} | p$. The legitimate prover has her own proof with the tuple $(x, c, d, A, y, b_1, b_2)$ such that $[\mathcal{P}_P(K_P), \mathcal{V}_V()](\kappa, P_P) \rightarrow 1$ or $A y^c b_2 = g^{a_1} h^{a_2} | p$. Since $g^{a'_1} h^{a'_2} / d' = g^{a_1} h^{a_2} / d | p$ or $g^{\bar{a}'_1 + \alpha \bar{a}_2} = g^{a_1 + \alpha a_2} | p$, where $\bar{a}'_i \leftarrow a'_i - r'_i$ and $\bar{a}_i \leftarrow a_i - r_i$, *TTP* can compute α ,

$$\alpha = (\bar{a}'_1 - \bar{a}_1)(\bar{a}'_2 - \bar{a}_2)^{-1} | q.$$

This is because $\bar{a}'_2 \neq \bar{a}_2$. Note that, since $\bar{a}'_2$ and $\bar{a}_2$ are independent of the proof commitment with respect to r_1 and r_2 , α will never be found in the case of two legitimate signatures (because $\bar{a}'_2 = \bar{a}_2$).

For convenience, let us now extend definitions given in Definitions 1 and 2 in Section 2:

Definition 3. (*Fail-Stop Knowledge Proof – Extension to Definition 1*) The fail-stop knowledge proof of discrete log, $\log_g y$, described in the above protocol, is represented by $\kappa \leftarrow FSP(K_P, P_P, A, d)(x, y)$ or simply $\kappa \leftarrow FSK(x, y)$.

Definition 4. (*Fail-Stop Signature Knowledge Proof – Extension to Definition 2*) Let $x \in \mathcal{Z}_q$ be the secret key and $y = g^x|p$ be the commitment to x . Let m be the message to be signed. The fail-stop signature knowledge is the tuple $(a_1, a_2, c) \in \mathcal{Z}_q^3$ for $a_1 = r_1 + x_2 + cx x_1$, $a_2 = r_2 + x_3 + cx x_4$ and $c = H(m, g, y, h, b_1, b_2, A, dy^c b_2)$, defined as $\sigma \leftarrow FSSK(x, y)(m)$.

3.2 Fail-Stop Signature Knowledge Proofs for Discrete Logarithm Equality

Consider $g_1, g_2 \in \mathcal{Z}_p^*$. Let $x \in \mathcal{Z}_q$ be the secret to be proved by the prover and $y_1 = b_1^x|p$ and $y_2 = b_1'^x|p$ be her commitments to the secret. The prover chooses $x_1, \dots, x_4$ from $\mathcal{Z}_q$ at random as her secret key and computes $b_1 \leftarrow g_1^{x_1} h_1^{x_4}|p$, $b_1' \leftarrow g_1^{x_1} h_2^{x_4}|p$, $b_2 \leftarrow g_1^{x_2} h_1^{x_3}|p$, and $b_2' \leftarrow g_1^{x_2} h_2^{x_3}|p$. Let $Cert \leftarrow (b_1, b_1', b_2, b_2')$ be the prover's public key certificate signed by the TTP. The key for the prover is defined as $K \leftarrow (K_P, P_P)$ for $K_P \leftarrow (x, x_1, \dots, x_4)$ and $P_P \leftarrow (y_1, y_2, b_1, b_1', b_2, b_2')$.

The protocol is as follows:

- Prover:
 - Selects $r_1, r_2 \in_R \mathcal{Z}_q$ and computes $d_1 = g_1^{r_1} h_2^{r_2}|p$ and $d_2 = g_2^{r_1} h_2^{r_2}|p$.
 - Calculates $c = H(m, g_1, g_2, h_1, h_2, y_1, y_2, b_1, b_2, b_1', b_2', d_1 y_1^c b_2, d_2 y_2^c b_2')$, where $h_1 = g_1^x|p$ and $h_2 = g_2^x|p$; both are determined by the TTP.
 - Then calculates $a_1 = r_1 + x_2 + x_1 cx$ and $a_2 = r_2 + x_3 + x_4 cx$ for $A \leftarrow (a_1, a_2) \in \mathcal{Z}_q \times \mathcal{Z}_q$.
- Verifier:
 - Checks the hash function.
 - Checks $d_1 y_1^c b_2 \stackrel{?}{=} g_1^{a_1} h_1^{a_2}|p$ and $d_2 y_2^c b_2' \stackrel{?}{=} g_2^{a_1} h_2^{a_2}|p$.

For the correctness of this protocol, readers are referred to the explanation given in the proceeding protocol for soundness and completeness.

To obtain a signature knowledge proof, we only need to modify the hash function and compute a different c . The definition below gives the detail.

Definition 5. (*Fail-Stop Chuam-Petersen Signature Knowledge for Equality of Discrete Logarithms*) Let $x \in \mathcal{Z}_q$ be the secret key, m be the message, $y_1 = b_1^x|p$ and $y_2 = b_1'^x|p$ be the public key, and the signature commitment be d_1, d_2 . The signature knowledge is the triplet $(a_1, a_2, c) \in \mathcal{Z}_q^3$ for $a_1 = r_1 + x_2 + x_1 cx$ and $a_2 = r_2 + x_3 + x_4 cx$ and $c = H(m, g_1, g_2, h_1, h_2, y_1, y_2, b_1, b_2, b_1', b_2', d_1 y_1^c b_2, d_2 y_2^c b_2')$. The Fail-Stop signature knowledge proof for equality of discrete logarithms is defined as $FSCP(x, g_1, g_2, y_1, y_2)(m)$.

4 Blind Fail-Stop Signature Knowledge

In order to study the blind fail-stop confirmer signatures, we give the blind version of fail-stop knowledge proofs and signature knowledge.

4.1 Blind Fail-Stop Signature Knowledge Proofs

Let x be the secret to be proved by the prover and $y = g^x|p$ be his commitment to the secret. Based on the bi-proof of two discrete logs[13], we can assume the base for the proof is b_1 . That is, $y = b_1^x|p$. The protocol is described below:

- Signer: Selects $r_1, r_2 \in_R \mathcal{Z}_q$ and computes $d = g^{r_1} h^{r_2}|p$, and sends d to the Recipient.
- Recipient: Selects $\gamma, \theta \in_R \mathcal{Z}_q$, computes $d' \leftarrow (db_2)^\theta b_1^{-1} g^\gamma|p$, $c = H(m, g, h, y, b_1, b_2, d')$, and $c' \leftarrow c/\theta$, and then sends c' to the signer.
- Signer: Computes $a_1 = r_1 + x_2 + x_1 c'x$ and $a_2 = r_2 + x_3 + x_4 c'x$ for $A \leftarrow (a_1, a_2) \in \mathcal{Z}_q \times \mathcal{Z}_q$, and then sends A to the recipient.
- Recipient: Computes $a'_1 \leftarrow a_1 \theta + \gamma$ and $a'_2 \leftarrow a_2 \theta + \gamma$. $A' \leftarrow (a'_1, a'_2)$.

The blind signature knowledge consists of A', c, b_1, b_2, d' . The verifier can validate it by checking the hash function and $d'y^c b_2 \stackrel{?}{=} g^{a'_1} h^{a'_2}|p$.

The completeness of the protocol is obvious. For the soundness of the protocol, we can see the following properties:

- Blindness: The signer/prover has no knowledge on the resulting signature knowledge, since he signed only c' . However, given the signature knowledge tuple (A', c, b_1, b_2, d') , both signer and recipient can be sure of its authenticity.
- Unforgeability: Although the signer has not seen c , in the verification of c , the verifier checks the details of the hash function and makes sure that d' has been embedded in it. Therefore, d' cannot be fault.
- Fail-stop: The signature doublet (c, A') has a form as a normal fail-stop confirmer signature, therefore Lemma 1 can still be applied.

Definition 6. (*Blind Fail-Stop Signature Knowledge Proof*) Let m be the message to be signed. The blind fail-stop signature knowledge proof is the pair (A', c') $\in \mathcal{Z}_q^3$ defined as $BFSSK(x, y)(m)$.

4.2 Blind Fail-Stop Signature Knowledge for Discrete Logarithm Equality

The setup of keys is the same as that for a normal *FSCP*. The only difference is that c is unknown to the signer.

- Signer: Selects $r_1, r_2 \in_R \mathcal{Z}_q$ and computes $d_1 = g_1^{r_1} h_1^{r_2}|p$ and $d_2 = g_2^{r_1} h_2^{r_2}|p$, and sends d_1, d_2 to the Recipient.
- Recipient: Selects $\gamma, \theta \in_R \mathcal{Z}_q$. computes $d'_1 \leftarrow (db_2)^\theta b_1^{-1} g_1^\gamma$, $d'_2 \leftarrow (db_2)^\theta b_1'^{-1} g_2^\gamma$, $c = H(m, g_1, g_2, h_1, h_2, y_1, y_2, b_1, b_2, b'_1, b'_2, d'_1 y_1^c b_2, d'_2 y_2^c b'_2)$, and $c' \leftarrow c/\theta$, and then sends c' to the signer.
- Signer: Computes $a_1 = r_1 + x_2 + x_1 c'x$ and $a_2 = r_2 + x_3 + x_4 c'x$ for $A \leftarrow (a_1, a_2) \in \mathcal{Z}_q \times \mathcal{Z}_q$, and then sends A to the receiver.

– Recipient: Computes $a'_1 \leftarrow a_1\theta + \gamma$ and $a'_2 \leftarrow a_2\theta + \gamma$. $A' \leftarrow (a'_1, a'_2)$.

The blind signature knowledge consists of A', c, b_1, b_2, d' . The verifier can validate it by checking the hash function and $d'_1 y_1^c b_2 \stackrel{?}{=} g_1^{a'_1} h_1^{a'_2} |p$ and $d'_2 y_2^c b'_2 \stackrel{?}{=} g_2^{a'_1} h_2^{a'_2} |p$. The correctness of this protocol can be seen from the discussions for *BFSK* and *FSCP*.

Definition 7. (*Blind Fail-Stop Chuam-Petersen Signature Knowledge for Equality of Discrete Logarithms*) Let $x \in \mathcal{Z}_q$ be the secret key and m be the message. Let $g_1, g_2 \in \mathcal{Z}_p^*$ and $y_1 = b_1^x |p$ and $y_2 = b_1^x |p$ be the public key, and the signature commitment be d'_1, d'_2 . The signature knowledge is the pair $(A', c) \in \mathcal{Z}_q^3$ for $c = H(m, g_1, g_2, h_1, h_2, y_1, y_2, b_1, b_2, b'_1, b'_2, d'_1 y_1^c b_2, d'_2 y_2^c b'_2)$, defined as *BFSK*(x, g_1, g_2, y_1, y_2)(m).

5 Fail-Stop Confirmer Signatures

Based on the technique we developed in Sections 3 and 4, we can easily construct a fail-stop confirmer signature scheme and its blind version. We will adopt Chaum's confirmer signature scheme.

5.1 Signing Protocol

Consider $g \in \mathcal{Z}_p^*$, $x_s, \alpha \in \mathcal{Z}_q$, $y_s = g^{x_s} |p$, and $f = g^{x_c} |p$, where x_c is the confirmer's secret key. Assume that the signer has a secret key $K_S \leftarrow (x_s, x_1, \dots, x_4)$ and its corresponding public key $P_S \leftarrow (y_s, b_1, b'_1, b_2, b'_2)$.

The signing protocol is as follows:

Step 1: The signer constructs the signature:

- Selects $r \in_R \mathcal{Z}_q$. $\bar{K}_S \leftarrow (r, x_1, \dots, x_4)$.
- Computes $u = g^r |p$ and $v = f^r |p$. $\bar{P}_S \leftarrow (u, v)$
- Computes the fail-stop Chuam-Petersen signature knowledge for discrete logs, $\log_g u = \log_f v$, on m . That is, $\sigma \leftarrow FSCP(r, g, h, u, v)(m)$.
- Computes the fail-stop signature knowledge on σ : $\bar{\sigma} \leftarrow FSSK(x_s, y_s)(\sigma)$.
- Then sends $u, v, \sigma, \bar{\sigma}$ to the verifier.

Step 2: The verifier verifies the signature by checking

$$[\mathcal{S}_S(\bar{K}_S), \mathcal{V}_V()](m, \sigma, \bar{P}_S \cap P_S, f) \rightarrow 1.$$

$$[\mathcal{S}_S(K_S), \mathcal{V}_V()](\sigma, \bar{\sigma}, P_S,) \rightarrow 1.$$

Completeness: It is obvious that following the correct steps the verifier can obtain the correct form of the signature. $[\mathcal{S}_S(\bar{K}_S), \mathcal{V}_V()](m, \sigma, \bar{P}_S, f) \rightarrow 1$ ensures that the signer knows r and its commitment (u, v) to r . The verifier is sure that the confirmer's commitment f has been embedded in σ . With $[\mathcal{P}_S(K_S), \mathcal{V}_V()](\sigma, \bar{\sigma}, P_S) \rightarrow 1$, the verifier ensures that the message is properly signed.

Soundness: We need to prove that the signer cannot cheat during the computation of σ and $\bar{\sigma}$.

- Unforgeability: There exists no a polynomial time algorithm which, on input (P_S, y_s, f) from the given key generators, outputs with non-negligible probability a correct confirmation signature knowledge, $(\sigma', \bar{\sigma}', m')$. This property holds for the situation that there is no an adversary who has sufficient computational power that can solve discrete log problems. The fail-stop property handles any adversary who is powerful enough to compute a discrete log.
- Fail-stop property: Both signature knowledge proofs, σ and $\bar{\sigma}$, are fail-stop. That is, the system allows the existence of some no polynomial time algorithms, which, on the given P_S for σ , output a $P'_S \leftarrow (y'_s, b_1, b'_1, b_2, b'_2)$, where (b_1, b'_1, b_2, b'_2) maps to $(\hat{x}_i, \hat{x}'_i) \not\rightarrow (x_i, x'_i)$. However, such forgery can be stopped with probability $1 - 1/q$. The similar situation is applied to $\bar{\sigma}$.

5.2 Confirmation Protocol

We now show how the confirmer confirms the confirmer signature knowledge by the signer. Basically, the confirmer needs only to proof his knowledge on the discrete log, $x_c \leftarrow \log_u v$. To make the proof fail-stop, we need to apply the fail-stop discrete log proof given in Section 3. The proof is defined as $\kappa_c \leftarrow FSK(\alpha, v)(\sigma, \bar{\sigma})$. The verifier checks $[\mathcal{P}_C(K_C), \mathcal{V}_V()](\kappa_c, (u, v))$ and makes sure the output is 1. The properties of the confirmation protocol follows these given in Section 3.

5.3 Blind Fail-Stop Confirmer Signatures

Blind signatures are useful for many applications in e-commerce. Therefore, it is worthwhile to consider the methods for blinding our Fail-Stop confirmer signatures. The blind fail-stop confirmer signature scheme has a similar protocol as the non-blind version. Only modification needed is to change the *FSCP* and *FSSK* to *BFSCP* and *BFSSK*. We omit the presentation of the protocols in this paper.

6 Conclusion

We have presented a security enhanced confirmer digital signature scheme, which prevents a powerful adversary from forging a signature. Our system is based on the theory of fail-stop knowledge proofs on discrete logs and signature knowledge developed in the paper; we have shown that the scheme is unforgeable and fail-stop. We have also constructed the blind version for the scheme, which has potential applications in electronic commerce. An important contributions in this paper is fail-stop zero knowledge proofs on discrete logarithms. This new concept makes such zero knowledge proof schemes more secure.

References

1. D. Chaum, "Zero-knowledge undeniable signatures," in *Advances in Cryptology, Proc. EUROCRYPT 91*, LNCS 547, pp. 458–464, Springer-Verlag, 1991.

2. D. Chaum, "Designated confirmer signatures," in *Advances in Cryptology, Proc. EUROCRYPT 94*, LNCS 950, pp. 86–91, Springer-Verlag, 1994.
3. T. Okamoto, "Designated confirmer signatures and public key encryption," in *Advances in Cryptology, Proc. CRYPTO 94*, LNCS 839, pp. 61–74, Springer-Verlag, Berlin, 1994.
4. M. Michels and M. Stadler, "Generic constructions for secure and efficient confirmer signature schemes," in *Advances in Cryptology, Proc. EUROCRYPT 98*, LNCS 1403, pp. 405–421, Springer-Verlag, 1999.
5. K. Q. Nguyen, Y. Mu, and V. Varadharajan, "Undeniable confirmer signatures," in *International Security Workshop (ISW'99)*, LNCS, pp. 235–246, Springer-Verlag, 1999.
6. E. van Heyst and T. P. Pedersen, "How to make efficient fail-stop signatures," in *Advances in Cryptology, Proc. EUROCRYPT 92*, LNCS 658, pp. 366–377, Springer-Verlag, Berlin, 1993.
7. E. van Heyst, T. P. Pedersen, and B. Pfitzmann, "New constructions of fail-stop signatures and low bounds," in *Advances in Cryptology, Proc. CRYPTO 92*, LNCS 740, pp. 15–30, Springer-Verlag, Berlin, 1993.
8. I. B. Damgard, T. P. Pedersen, and B. Pfitzmann, "On the existence of statistically hiding bit schemes and fail stop signatures," in *Advances in Cryptology, Proc. CRYPTO 93*, LNCS 773, pp. 250–265, Springer-Verlag, Berlin, 1993.
9. N. Baric and B. Pfitzmann, "Collision-free accumulators and fail-stop signatures schemes without trees," in *Advances in cryptology - EUROCRYPT'97, Lecture Notes in Computer Science 1233*, pp. 480–494, Springer-Verlag, Berlin, 1997.
10. D. Chaum, J.-H. Evertse, and J. V. de Graaf, "An improved protocol for demonstrating possession of discrete logarithms and some generalizations," in *Advances in Cryptology, Proc. EUROCRYPT 87*, LNCS 304, pp. 128–141, Springer-Verlag, 1988.
11. J. Camenisch, "Efficient and generalized group signatures," in *Advances in cryptology - EUROCRYPT'97, Lecture Notes in Computer Science 1233*, pp. 465–479, Springer-Verlag, Berlin, 1997.
12. D. R. Stinson, *Cryptography, Theory and Practice*. CRC Press, 1995.
13. A. Fujioka, T. Okamoto, and K. Ohta, "Interactive bi-proof systems and undeniable signature schemes," in *Advances in Cryptology, Proc. EUROCRYPT 91*, LNCS 547, pp. 243–256, Springer-Verlag, Berlin, 1991.

An Extremely Small and Efficient Identification Scheme

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. We present a new identification scheme which is based on Legendre symbols modulo a certain hidden prime and which is naturally suited for low power, low memory applications.

1 Overview

One of the most desirable cryptographic functions is a secure, small, zero-knowledge public-key identification scheme. The applications are many and obvious – even the single application of credit/smart-card security is enough to stimulate research. In this paper, we present a scheme that requires extremely little computing power to perform a verification and to which we refer as FLIP (Fast Legendre Identification Protocol). Our scheme is unbalanced by design: the party proving his/her identity needs almost no computing power, while the party to whom the identity is being proved needs only a very small amount.

Our scheme is based on the assumption that integer factorization is a “hard problem.” In fact, we believe that the *only* feasible attack on our scheme is via the factorization of a certain modulus M , hence the scheme is secure provided that M is reasonably large. In contrast to (say) **RSA** based signatures, our scheme offers the advantage that time consuming computation modulo M is required only for the *verifier* (it is not unreasonable to assume the *verifier* to be more powerful than the *prover*). This lends itself well to the credit/smart-card & bank paradigm.

We have conducted some preliminary tests using the interpreted number theory system PARI-GP. We find that FLIP (at high security) performs an identification as fast or faster than any other identification scheme of which we are aware, although key creation in FLIP is slower than in some other schemes, see [8].

We remark that although our scheme uses constructions similar to the Feige–Fiat–Shamir scheme, see [10], both schemes seem to be independent and rely on the hardness of different number theoretic problems (although the assumption of the hardness of integer factorization is common to both of them).

2 FLIP Description

Our construction is based on properties of the *Legendre* and *Jacobi* symbols.

We recall that given a prime p the Legendre symbol of a with $\gcd(a, p) = 1$ is defined as

$$\begin{cases} 1, & \text{if the congruence } a \equiv x^2 \pmod{p} \text{ is solvable,} \\ -1, & \text{otherwise.} \end{cases}$$

The Jacobi symbol modulo an odd integer k is defined as multiplicative extension of the Legendre symbol. That is,

$$\left(\frac{a}{k}\right) = \prod_{\nu=1}^s \left(\frac{a}{p_\nu}\right)^{e_\nu}$$

where

$$k = p_1^{e_1} \dots p_s^{e_s}$$

is the prime factorization of k .

We also recall the following basic properties of the Jacobi symbol (see Section 5.8 of [3]) which hold for any odd integer k and arbitrary integers l and r :

- $\left(\frac{2}{k}\right) = (-1)^{(k^2-1)/8}$;
- $\left(\frac{lr}{k}\right) = \left(\frac{l}{k}\right) \left(\frac{r}{k}\right)$;
- $\left(\frac{l}{k}\right) = \left(\frac{r}{k}\right)$ if $l \equiv r \pmod{k}$;
- $\left(\frac{l}{k}\right) = \left(\frac{k}{l}\right) (-1)^{(l-1)(k-1)/4}$, if l is odd.

The above properties provide very fast algorithms for computing Jacobi symbols, and thus guarantee the efficiency of our scheme.

FLIP has two formal security parameters, integers n and k .

For purposes of this paper, we will assume that *Irina* is proving her identity to the verifier *Victor*.

To create the signature *Irina* uses the following algorithm:

FLIP initial set-up and key construction

Step 1

Irina chooses two n -bit prime numbers p and r , and computes the product $M = pr$.

Step 2

Irina selects at random k relatively prime $2n$ -bit integers a_j , computes the Legendre symbols

$$\alpha_j = \left(\frac{a_j}{p} \right), \quad j = 1, \dots, k,$$

and checks that at least one $\alpha_j = -1$.

Step 3

Irina publishes as her public key the product M and the collection of k pairs (a_j, α_j) , $j = 1, \dots, k$.

Step 4

Irina discards the value of r and retains as her private key the prime p .

The verification protocol has another security parameter which is a non-negative integer l .

The FLIP verification sequence**Step 1**

Victor chooses l random $2n$ -bit integers $s_1, \dots, s_l$ and l subsets of the set $\{a_1, \dots, a_k\}$, and for each subset, he computes the product of s_i^2 and the selected integers modulo M . In other words, *Victor* chooses l sets of k random bits $e_{ij} = 0, 1$, and computes for $i = 1, \dots, l$

$$C_i \equiv s_i^2 \prod_{j=1}^k a_j^{e_{ij}} \pmod{M}, \quad 0 \leq C_i \leq M - 1.$$

Step 2

Victor transmits the l numbers C_i , $i = 1, \dots, l$.

Step 3

Irina computes and transmits the l Legendre symbols

$$\vartheta_i = \left(\frac{C_i}{p} \right), \quad i = 1, \dots, l.$$

Step 4

Victor verifies each of the l Legendre symbols transmitted by *Irina* is correct. That is, he verifies that

$$\vartheta_i = \prod_{j=1}^k \alpha_j^{e_{ij}}, \quad i = 1, \dots, l.$$

We first note that in terms of the parameters n , k and l , we have the following properties:

- the bit size of the private key length is n

- the bit size of the public key length is $2n + k(n + 1)$;
- the total number of bits transmitted is $(2n + 1)l$.

We remark that the relatively prime numbers $a_1, \dots, a_k$ need not be chosen by *Irina*. They can be globally available or even produced by *Victor*. In either of these cases, they are not strictly speaking a part of the public key, and the bit length of the public key drops to $2n + k$. In any case, the generation of the a_i is not time consuming. In fact even if one decides to select k random $2n$ -bit prime numbers this can be done efficiently by selecting random integers in the interval $[2^{n-1}, 2^n - 1]$ and testing them for primality. Classical density results about the distribution of prime numbers and primality testing algorithms (see [3, 5, 6]) guarantee the efficiency of this procedure. The condition that at least one of $a_1, \dots, a_k$ is not a quadratic residue modulo p is very easy to satisfy as well, for example, by selecting a_1 with this property.

It is also useful to recall that each computation of a Legendre symbol involved in this scheme takes $O(n^2)$ bit operations; see Theorem 5.9.3 of [3] or Section 1.4 of [5]. Each multiplication modulo M takes $O(n^2)$ bit operations if one uses naive arithmetic and $O(n \log n)$ bit operations if one uses fast arithmetic, see Theorems 7.8 and 8.5 of [1] or Theorem 8.24 and Corollary 9.9 of [6].

3 Security Analysis

It is obvious that the probability of impersonating a valid private key (that is, the probability of a correct guess of l individual Legendre symbols) is 2^{-l} . This is an “on-line” attack, and for attacks of this type, it is common to request the 2^{40} level of security. Thus the choice $l = 40$ will be satisfactory.

The probability that the same product will be used twice during N rounds of verification, thus allowing an attacker to collect and re-use *Irina*’s replies, is

$$P_{l,k,N} = 1 - (1 - 2^{-k})^{Nl}.$$

In particular $P_{l,k,N} \sim Nl2^{-k}$ if $Nl2^{-k} \sim 0$. This is an “off-line” attack, and for attacks of this type, it is common to request the 2^{80} level of security. Assuming that $l = 40$ and that $N = 10000$ identification rounds are to be made, one can easily verify that the choice $k = 99$ guarantees

$$P_{l,k,N} \leq 2^{-80}.$$

An “off-line” brute force attack (that is, correctly guessing a valid private key) would succeed with probability 2^{-k} .

One can also apply the above scheme with $s_1 = \dots = s_l = 1$. However in this case a more sophisticated attack can be used. An attacker can precompute the products

$$\prod_{j=1}^{\lfloor k/2 \rfloor} a_j^{f_j} \quad \text{and} \quad \prod_{i=\lfloor k/2 \rfloor + 1}^k a_i^{g_i}$$

for all binary vectors $(f_1, \dots, f_{\lfloor k/2 \rfloor})$ and $(g_{\lfloor k/2 \rfloor + 1}, \dots, g_k)$ and then try to find a representation of the challenges C_i , $i = 1, \dots, l$, by looking at the precomputed values. This “meet in the middle” attack requires of order $2^{k/2}$ operations and the same amount of memory, so it is not likely to be efficient. In any case, using random squares in the computation of the challenges rules out this attack completely.

Another possible attack is via the known values of Legendre symbols. In theory, if one knows the Legendre symbols modulo p of the first $O(\log^2 p)$ integers, then this is enough to identify p uniquely (see [7]), but no one has been able to produce an efficient algorithm to accomplish this identification; indeed, the security schemes defined in [2] are based on the intractability of this problem. Note that in our scheme *Irina* does not verify that the numbers C_i , $i = 1, \dots, l$, for which she is supposed to compute Legendre symbols are valid products of the integers a_j , $j = 1, \dots, k$, used to construct the public key (indeed, such a verification would be infeasible). Thus, the attacker can force *Irina* to compute the Legendre symbol modulo p of any integer C . However, we believe that identifying the prime number p from the values of Legendre symbols modulo p of integers is completely infeasible.

Of course the attacker is able to compute

$$\left(\frac{C}{r}\right) = \left(\frac{C}{M}\right) \left(\frac{C}{p}\right)$$

for any C as well, but the same arguments as above apply to this as well.

As one of the advantages of our scheme we note that it is an *honest verifier zero-knowledge* scheme. That is, an honest verifier, who uses only “legitimate” challenges C_i , i.e., challenges of the form

$$C_i \equiv s_i^2 \prod_{j=1}^k a_j^{e_{ij}} \pmod{M}, \quad 0 \leq C_i \leq M-1,$$

$i = 1, \dots, l$, does not obtain any new information from the prover.

Finally, one could successfully attack this scheme by factoring either M or finding the representation of C_i as a product of powers of a_j . That is, by finding representations

$$C_i \equiv \prod_{j=1}^k a_j^{x_{ij}} \pmod{M}, \quad i = 1, \dots, l,$$

with integer x_{ij} . However, it is easy to see that the latter problem is not easier than the discrete logarithm problem. Indeed, even if $a_1, \dots, a_k$ belong to a cyclic group $\mathcal{G}$ modulo M and even if representations $a_j \equiv g^{d_j} \pmod{M}$, $j = 1, \dots, l$, are known, where g is a generator of $\mathcal{G}$, then finding a representation

$$C \equiv \prod_{j=1}^k a_j^{x_j} \equiv g^{x_1 d_1 + \dots + x_k d_k} \pmod{M}$$

is no easier than the general discrete logarithm problem modulo M . However, it has been shown in [9] that the discrete logarithm problem modulo a composite $M = pr$ (or even the possibly easier problem of breaking the Diffie–Hellman scheme) is as hard as factoring M ; see also [4]. In particular, the prime numbers p and r should be selected to avoid the all known “short-cuts” in the integer factorization of $M = pr$ and in solving the discrete logarithm problem modulo M . Some conditions of this kind have been described in [9].

4 Possible Modifications

Instead of using quadratic characters one can use characters of higher orders, for example bi-quadratic characters. In this case a smaller value of l can be selected for the verification procedure, thus reducing the number of bits exchanged. For example, using bi-quadratic characters, one can use a value of l that is twice as small in order to provide the same level of security. More generally, characters of order d reduce this value by approximately $d/2$ times. On the other hand, the computational cost of computing higher order characters grows rather quickly with d . Nevertheless, our preliminary computational experiments have confirmed that bi-quadratic characters can be incorporated rather efficiently in this scheme.

Another possible modification may help to hide the values ϑ_i , $i = 1, \dots, l$. In order to do so, *Irina* and *Victor* select some large integer weights w_i , $i = 1, \dots, l$, (cooperatively, say each of them provides half of the bits of each element). Then *Irina* sends the sum $W = \vartheta_1 w_1 + \dots + \vartheta_l w_l$ which can be verified by *Victor*. However finding the values of $\vartheta_1, \dots, \vartheta_l$ from the value of W is equivalent to the knapsack problem which is known to be **NP**-complete. In fact, there is no need to select the weights w_i , $i = 1, \dots, l$, for each round. They can be some initially agreed upon functions of ϑ_i , $i = 1, \dots, l$. Moreover, if l is too small to guarantee the security of the knapsack problem, then *Irina* and *Victor* may use more weights w_i , $i = 1, \dots, L$, with some $L \geq l$ and compute the sum

$$W = B_1(\vartheta_1, \dots, \vartheta_l)w_1 + \dots + B_L(\vartheta_1, \dots, \vartheta_l)w_L$$

where B_i , $i = 1, \dots, L$, are some Boolean functions of l variables. Probably one can even use $B_i(\vartheta_1, \dots, \vartheta_l) = \vartheta_i$ for $i = 1, \dots, l$.

References

1. A. V. Aho, J. E. Hopcroft and J.D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, MA, 1975.
2. M. Anshel and D. Goldfeld, ‘Zeta functions, one-way functions, and pseudorandom number generators’, *Duke Math. J.*, **88** (1997), 371–390.
3. E. Bach and J. Shallit, *Algorithmic number theory*, MIT Press, 1996.
4. E. Biham, D. Boneh and O. Reingold, ‘Breaking generalized Diffie–Hellman modulo a composite is not weaker than factoring’, *Inform. Proc. Letters*, **70** (1999), 83–87.

5. H. Cohen *A course in Computational Algebraic Number Theory*, Springer-Verlag, Berlin, 1997.
6. J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, Cambridge, 1999.
7. D. Goldfeld and J. Hoffstein, 'On the number of Fourier coefficients that determine a modular form', *Contemp. Math.*, **143** (1993), 385–393.
8. J. Hoffstein, D. Lieman and J. Silverman, 'Polynomial rings and efficient public key authentication', *Proc. the Intern. Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, M. Blum and C.H. Lee, eds., City University of Hong Kong Press (to appear).
9. K. S. McCurley, 'A key distribution system equivalent to factoring', *J. Cryptology*, **1** (1988), 95–105.
10. A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.

An Anonymous Electronic Bidding Protocol Based on a New Convertible Group Signature Scheme

– On the importance of hiding the winner’s identity against bid-rigging –

Kouichi Sakurai¹ and Shingo Miyazaki²

¹ Graduate School of Information and Electrical Engineering, Kyushu University,
Hakozaki, Fukuoka, 812-8581, JAPAN.

sakurai@csce.kyushu-u.ac.jp

² System Integration Technology Center, Toshiba Corporation,
3-22, Katamachi, Fuchu-shi, Tokyo 183-8512, JAPAN

shingo@sitc.toshiba.co.jp

Abstract. We propose an electronic auction scheme satisfying that (i) a group of colluding bidders cannot control the contract price arbitrarily, (ii) all prices of bidders except the winner are never revealed to anyone (even to the auction house), and (iii) only the auction house recognizes the winner’s identity, while the other losers can verify the fact that the winner belongs to the advanced registered group.

Our scheme does not rely on an anonymous channel nor a trusted third center. Our system can be implemented only with the public-key technology and cryptographic hash functions and all bidders can verify the validity of process for determining a winner via a public bulletin board. To achieve stronger anonymity of the winner, we develop a new variant of convertible undeniable group signatures. In our designed signature scheme, the convertibility has two phases: one is convertible from *on-line* to *off-line* in verification-stage, and the other is convertible from individual to group.

Key words: *electronic auction, anonymous bidding, bid rigging, bidder’s privacy, non-repudiation, undeniable signature, group signature, bulletin board*

1 Introduction

A digital auction scheme is viewed as a set of electronic protocols which allow a collection of bidders to buy a thing at an auction with the low price as possible, while a seller puts a thing up for auction and wants the bidders to buy its own exhibit with the highest price possible. With the use of recent development public-key cryptographic techniques, we propose a digital auction scheme satisfying the following requirements.

Secrecy of bidding price:

All bidding prices except the contract price are *never* revealed to anybody including the auctioneer.

Validity of the successful bid:

The price of the successful bid is the highest one among all bidding prices.

Fairness:

No bidder can tender with an advantageous condition than the other bidders.

Anonymity of the winner:

Only the auction house recognizes the winner's identity, while the other losers can verify the fact that the winner belongs to the advanced registered group.

Our main contribution of this paper is that our scheme can be implemented only with a bulletin board, which is a broadcast channel with memory that can be observed and read by all parties (e.g. the WEB), and does not require anonymous channels nor trusted third authorities.

1.1 Overview of Our Approach

The base approach of our scheme is a bidding down strategy adding to a sealed-bid auction:

Bidding: Each bidder j chooses a bidding price w_k from a set of published m prices $(w_1, w_2, \dots, w_m)$ and encrypts his bidding with his private-key to get his sealed bid $\mathcal{E}_j(w_k)$. The bidder publishes $\mathcal{E}_j(w_k)$.

Auction: The auctioneer sets the highest price w_m and asks each bidder if it tenders with the price via a bulletin board. For a price w_k , the bidder with w_k reveals the decrypted w_k for his sealed bid $\mathcal{E}_j(w_k)$.

A problem of this basic scheme is that a cheating bidder does not open his sealed bid $\tilde{w}$ even though the auctioneer asks whether the content of sealed bid on the bulletin board is the (current) highest price $\tilde{w}$. This bidder's cheating could break the validity of the successful bid: the price of the successful bid must be the highest among all bidding prices.

We overcome this problem of the invalidity of the successful bid by adding non-repudiation mechanisms to the basic scheme. For achieving non-repudiation mechanisms our proposed scheme adapts undeniable signatures [CvA89, Cha90]. Our novel modification of undeniable scheme is to use the undeniable signatures without signed messages.

1.2 Related Work

Electronic auction protocols are already executed on the WEB [ebay, Yah], and various protocols are investigated in cryptographic literatures [Yam88, IMI94, IM95, NWFK95, FR96, Kud98, KN98, HTK98, KHT98, SKFN98].

The similar type of digital auction discussed in this paper was investigated in [IMI94,IM95,NWFK95,Kud98,KN98,HTK98,KHT98,SKFN98], which are different from ours with respect not only to each achieved goal but also to applied cryptographic assumptions and required physical tools.

Imamura, Matsumoto, and Imai [IMI94] initiated the approach for fighting bid-rigging by using anonymous bidding. The scheme proposed in [NWFK95] furthermore considered both the issue of bid-rigging and the protection of bidders' privacy by assuming the existence of anonymous channels. We note that, there are some attempts to implement anonymous channels [RD97,SGR97], however, anonymous channels are not yet considered as matured tools as public-key based secret message transfer.

Kudo [Kud98] proposed an electronic sealed-bid auction protocol with public key cryptography. In this scheme, each bidder's selection is not directly revealed to anybody thanks to the pseudo-identity of bidders. Note that this scheme assumes that the registration authorities are distributed. Sako [Sak99] presented a bulletin-board based action protocol that hides losing bids. This is very practical, however, it still requires the trust of the third authority (auctioneer).

The scheme by Kikuchi and Nakanishi [KN98] and ours protect bidder privacy with cryptographic techniques, which require a computational assumption but no physical one. A difference from ours is that Kikuchi-Nakanishi's scheme [KN98] uses the multiparty protocol based on Shamir's secret sharing scheme for distributing bidder's identity to multiple auction-houses.

We should note that our proposed scheme requires no anonymous channels nor trusted third authorities, and we can realize our system over the WEB as a candidate of an implementable bulletin board only with the current public-key technology.

1.3 Our Contribution

The scheme by Nakanishi et al. [NWFK95] applied undeniable signature protocols to non-repudiation stage against bidder's cheating of opening the sealed bid [CvA89,Cha90].

Our Previous Scheme. We designed an electronic auction scheme [SM99] satisfying that (i) a group of colluding bidders cannot control the contract price arbitrarily, (ii) all prices of bidders except the winner are never revealed to anyone (even to the auction house).

Our previous scheme [SM99] uses an improved undeniable signature scheme developed by Michels and Stadler [MiSt97]. Their undeniable signature scheme is efficient, and the confirmation stage and the disavowal stage are done in a single protocol. Furthermore, their schemes are based on the discrete logarithm problem, that can be implemented over the elliptic curve crypto-system [Kob87, Mil85]. Furthermore, we use the undeniable signatures without signed messages, similar to the bit-commitment for sealed bids [SS99].

The Importance of Hiding the Winner's Identity. Our previous scheme [SM99] reveals the winner's identity for universal verifiability of the winning bid. However, it is important to hide the winner's identity from the other losers (except the auctioneer) for fighting against bid-rigging that could be coerced by a black-party.

Such a coercer would ask the bidders to submit his desired price and try to control the winning price. If the winner does not follow the coercer's request, and the winner's identity is disclosed, the coercer could take revenge on the winner.

While, in electronic voting schemes, fighting against coercion was already taken by the approach of receipt-freeness of voting [BT97a,SK95,Oka97], no previous work on digital auction systems discusses the winner's anonymity and the coercer's bid-rigging.

Our New Scheme. In this paper, we improve our previous scheme to achieve stronger anonymity of the winner: only the auction house recognizes the winner's identity, while the other losers can verify the fact that the winner belongs to the advanced registered group.

To this end, we develop a new variant of convertible undeniable group signatures. In our designed signature schemes, the convertibility has two phases: one is convertible from *on-line* to *off-line* in verification-stage, and the other is convertible from *individual* to *group*.

The idea of group signatures was introduced in [CvH91], which allows individual members of the group to sign messages on behalf of the group while remaining anonymous. We should note that our applied signature schemes is undeniable, i.e. the verification is done via on-line manner. In fact, our first scheme uses the Michels-Stadler's convertible undeniable signature scheme [MiSt97], which is convertible from *on-line* verification to *off-line* verification.

Our further requirement of the undeniable signature scheme is that the release of certain secret information by the signer (the winner) makes the signature convertible to off-line verification while preserving the group anonymity. Particularly, the verification by the auctioneer must be done with the winner's identity.

None of the previous convertible signature schemes [MiSt97,KPW96] satisfies our full requirements. Then, we originally develop our desired variant of convertible signatures.

2 Our New Convertible Group Signature Scheme

In this section, we describe the notations and then present our new convertible group signature scheme linked to the undeniable signature.

2.1 Notations

Let p and q be large primes which satisfy $p = 2q + 1$, α be a generator of a subgroup of order q , and $\mathcal{H}$ be a one-way hash function. Z_q denotes the ring of numbers modulus q and Z_q^* denotes the multiplicative group modulus q . Now, p , q , α and $\mathcal{H}$ are published as system parameters.

2.2 Protocols

The signer, Alice, has two secrets $x_1, x_2 \in Z_q^*$ and the corresponding keys, $y_1 = \alpha^{x_1} \pmod p$ and $y_2 = \alpha^{x_2} \pmod p$. Now, y_1 is known to only a manager, Bob while y_2 is published. Alice also has a group key x_G known to only members in her group. The public key P_G corresponding to x_G is published, where $P_G = \alpha^{x_G} \pmod p$.

In this situation, Alice generates her convertible undeniable signature and convertible group signature as follows.

For a message m , Alice selects two random numbers $k, \mu \in Z_q^*$ and computes both $r = \alpha^k \pmod p$ and $\lambda = \alpha^\mu \pmod p$. Then Alice calculates

$$\begin{aligned}\tilde{r} &= r^{x_2} \pmod p \\ \tilde{\lambda} &= \lambda^{x_2} \pmod p \\ c &= \mathcal{H}(m, \tilde{r}, \lambda, \tilde{\lambda}) \\ s &= k - cx_1 \pmod q \\ t &= \frac{\tilde{r}\mu + rk}{x_G + \mathcal{H}(m, \tilde{r}, \lambda)} \pmod q.\end{aligned}$$

Let $(\tilde{r}, s)$ be the convertible undeniable signature and $(\tilde{r}, t)$ be the convertible group signature. Nobody except Alice can verify both signatures without the help of Alice since r is never opened, and know the content of message by our approach proposed in [SM99] (we consider that the undeniable signature is still the ciphertext for the bid until it is converted to the standard digital signature). Alice sends $(\tilde{r}, \lambda, \tilde{\lambda}, s, t)$ to Bob, and publishes $(\tilde{r}, s, t)$.

When Alice sends the secret x_2 to Bob, Bob with y_1 can convert the undeniable signature to the standard digital signature. Firstly, Bob computes $\tilde{\lambda} = \lambda^{x_2} \pmod p$ and then checks the validity of signature via the verification formula:

$$\tilde{r} = (\alpha^s y_1^{\mathcal{H}(m, \tilde{r}, \lambda, \tilde{\lambda})})^{x_2} \pmod p.$$

Furthermore, the group signature is converted to the standard digital signature when Bob publishes $r = \tilde{r}^{x_2^{-1}} \pmod p$. Everyone can check the validity via the following verification formula:

$$\lambda^{\tilde{r}} r^r = (P_G \alpha^{\mathcal{H}(m, \tilde{r}, \lambda)})^t \pmod p$$

The direct application of this signature scheme to our auction system is vulnerable to an attack allowing the auctioneer to get the bidding price of any bidder, if the number of the candidates of submitted prices is few. Then, we revise the signature scheme above to achieve a kind of “semantic security” [GM84] (See Section 5).

3 Overview

3.1 Model

Our scheme has the following participants and a tool.

Auctioneer: The auctioneer calls sellers together and holds an auction. Then, it manages a bulletin board and writes various information on the board and presides the process of the auction.

Registration Authority: The registration authority registers bidders as the participants of auction and issues the certificate of bidder's public key.

Seller: The seller puts a thing up for auction and wants the bidders to buy its own exhibit with the high price as possible.

Bidders: The bidders tender for an article on exhibition and compete each other. Each bidder wants to buy a thing at an auction with the low price as possible.

Bulletin Board: Anybody can see all data filled in the bulletin board. Through all data on the bulletin board, the bidders and seller verify the correctness of process in the auction and the validity of determining a successful bidder.

3.2 Basic Protocol

Our scheme consists of the following basic protocols.

Registration: The registration authority issues the key certificate of each bidder.

Bidding: Each bidder chooses a bidding price w_k from a set of m prices $(w_1, w_2, \dots, w_m)$ and then makes both a convertible undeniable signature $\mathcal{US}(w_k)$ on w_k and a convertible group signature $\mathcal{GS}(w_k)$, where $w_1 < w_2 < \dots < w_m$. Then, the bidder publishes $\mathcal{US}(w_k)$ and $\mathcal{GS}(w_k)$ and the certificate *without attaching the message* w_k . Here, $\mathcal{US}(w_k)$ and $\mathcal{GS}(w_k)$ are considerable as ciphertexts of w_k .

Auction: The auctioneer sets the highest price w_m and asks each bidder if it tenders with the price via a bulletin board. Each bidder whose bidding price is *not* w_m must show that $\mathcal{US}(w_k)$ is *not* a signature on w_m via the *repudiation protocol*, without revealing its own bidding price w_k . The auctioneer lowers the price by degrees until a successful bidder appears, as verifying the validity of bidders' repudiation at every price. For a price w_k , the bidder with w_k must show that its own $\mathcal{US}(w_k)$ is the signature on w_k via the *confirmation protocol*. The auctioneer accepts the bidder as a successful bidder if and only if the proof is valid. The auctioneer converts $\mathcal{GS}(w_k)$ to the standard digital signature verifiable to anyone by using the information released in the phase of converting the undeniable signature to the ordinary digital signature.

If the repudiation protocol is omitted, a group of colluding bidders can control the contract price by abusing the denial of bidding. Nakanishi et al. [NWFK95] discussed this problem and produced a solution using the concept of undeniable signatures (See also our paper [SM99] describing the brief note on this).

The importance in this protocol is that the identity of winner is never published on the bulletin board. Each bidder except the auctioneer checks the validity of successful bidding with $\mathcal{GS}(w_k)$ converted to the verifiable signature. This verification requires the public key of winner's group including no information linked to the winner, i.e. the public key of the winner which is known to only the auctioneer.

3.3 Requirements

Our first scheme satisfies the following requirements.

Secrecy of bidding price:

All bidding prices except the contract price are not revealed to anybody including the auctioneer.

Validity of the successful bid:

The price of the successful bid is the highest among all bidding prices.

Fairness:

No bidder can tender with an advantageous condition than the other bidders.

Moreover, our improved scheme satisfies the following additional requirement.

Anonymity of the winner:

Only the auction house recognizes the winner's identity, while the other losers can verify the fact that the winner belongs to the advanced registered group.

4 An Anonymous Electronic Bidding Scheme

We propose a new auction protocol such that only the auction house recognizes the winner's identity, whereas the other losers can verify the fact that the winner exists in the advanced registered group.

[Registration]

The registration authority issues the secret key S_G to each member of the registered group ¹ via a secure channel. Let S_G be the group secret key and $P_G = \alpha^{S_G} \pmod{p}$ be its corresponding public-key.

Also let S_A be bidder A's individual private key and $P_A = \alpha^{S_A} \pmod{p}$ be its corresponding A's individual public-key, and $Cert_A$ is its certificate.

[Bidding Protocol]

Step.1: The auctioneer publishes m kinds of bidding prices. Let $(w_1, w_2, \dots, w_m)$ be a set of bidding prices, where $w_1 < w_2 < \dots < w_m$.

¹ The number of groups can be multiple.

Table 1. Released and published information

	Bidding stage	Repudiation	Opening stage
To Auctioneer	$P_G, (P_A, Cert_A, h, \tilde{r}, \lambda, \tilde{\lambda}, \sigma, t)$	—	x
On a bulletin-board	$P_G, (h, \tilde{r}, \sigma, t)$	β	λ, r

Step.2: Firstly, the bidder A chooses its own bidding price from $(w_1, w_2, \dots, w_m)$. Then the bidder A generates three random numbers $x, k, \mu \in Z_q^*$ and computes

$$\begin{aligned}
 h &= \alpha^x \pmod{p} \\
 r &= \alpha^k \pmod{p} \\
 \tilde{r} &= r^x \pmod{p} \\
 \lambda &= \alpha^\mu \pmod{p} \\
 \tilde{\lambda} &= \lambda^x \pmod{p} \\
 c &= \mathcal{H}(w_k, \tilde{r}, \lambda, \tilde{\lambda}) \\
 \sigma &= k - cS_A \pmod{q}.
 \end{aligned}$$

Then, the bidder A also calculates a group signature:

$$t = \frac{\tilde{r}\mu + rk}{x_G + \mathcal{H}(w_k, \tilde{r}, \lambda)} \pmod{q}$$

and sends $(P_A, Cert_A, h, \tilde{r}, \lambda, \tilde{\lambda}, \sigma, t)$ to the auctioneer. Remark that w_k should *not* be transferred at this step.

Step.3: The auctioneer writes $(h, \tilde{r}, \sigma, t)$ in the column of the concerned group on the bulletin board if and only if the validity of certificate $Cert_A$ is accepted.

Step.4: The auctioneer closes the request of bidding at a suitable time.

[Opening Protocol]

Step.1: The auctioneer sets and publishes the highest price w_m .

Step.2: The bidder who tendered with the price w_m shows the validity of its own bidding via the *confirmation protocol* proving the equality between the encrypted w_k and the called price w_m described in appendix A. If nobody offered its bid with w_m , each bidder must show that its own bidding price is *not* w_m , i.e. $(\tilde{r}, s)$ is *not* a signature on w_m , by the *repudiation protocol*, described in appendix A, with the following parameters:

$$\begin{aligned}
 y &= h \\
 z &= \tilde{r} \\
 \beta &= \alpha^\sigma P_A^{\mathcal{H}(w_m, \tilde{r}, \lambda, \tilde{\lambda})}.
 \end{aligned}$$

After checking the validity of bidders' repudiation, the auctioneer sets the second highest price w_{m-1} and repeats the above process until any successful bidder gives its name.

Step.3: For a price w_k , the bidder j who puts in its bid with w_k shows the auctioneer the validity of bidding by the *confirmation* protocol. The auctioneer considers the bidder as a candidate if and only if the check is successfully accepted. The bidder A is accepted as the successful bidder, when there is only a candidate.

In this case, the winner A should convert its *locked* group signature to the group signature verifiable to anyone. That is done by releasing its secret x to the auctioneer, which can also convert the undeniable signature of the winner to the digital signature. The process is as follows.

[3a]: The bidder A sends x to the auctioneer.

[3b]: The auctioneer verifies the validity of the (individual) signature by the following formula:

$$\tilde{r} = (\alpha^\sigma P_A^{\mathcal{H}(w_k, \tilde{r}, \lambda, \tilde{\lambda})})^x \pmod{p}$$

with the public-key P_A .

Furthermore, the auctioneer computes $r = \tilde{r}^{x^{-1}} \pmod{p}$ and verifies the group signature:

$$\tilde{\lambda} \tilde{r} r = (P_G \alpha^{\mathcal{H}(w_k, \tilde{r}, \lambda)})^{tx} \pmod{p}$$

[3c]: If both the individual signature and the group signature are correct, the auctioneer publishes (r, λ) .

[3d]: All bidders can verify the validity of the winning price with published data $(\lambda, r, \tilde{r}, t, w_k, P_G)$.

$$\lambda^{\tilde{r}} r^r = (P_G \alpha^{\mathcal{H}(w_k, \tilde{r}, \lambda)})^t \pmod{p}$$

5 A Secrecy Problem on Bidding Price and an Improved Signature Scheme

5.1 Disclosure of Bidding Prices

If the number of the candidates of submitted prices is few, our scheme described in section 4 might be vulnerable to the attack allowing the auctioneer to know the bidding price of any bidder. This attack is due to the similar problem of the deterministic (public-key) encryption scheme [GM84]: for all prices $w_i \in W = \{w_1, w_2, \dots, w_m\}$ and the bidding price w_k , the auctioneer searches w_i satisfying the verification formula one by one.

The concrete attacking algorithm is as follows.

1. For a price w_i , the auctioneer computes $R(w_i)$ with the data received from the target of bidder as follows.

$$\begin{aligned} R(w_i) &= \alpha^\sigma P_A^{c(w_i)} \pmod{p} \\ &= r P_A^{-c(w_k)} P_A^{c(w_i)} \pmod{p} \end{aligned}$$

Note that $R(w_i) = r$ if w_i equals w_k .

2. For $R(w_i)$ obtained at step 1, the auctioneer verifies the correspondence in the following formula.

$$(P_G \alpha^{\mathcal{H}(w_k, \tilde{r}, \lambda)})^t \stackrel{?}{=} \lambda^{\tilde{r}} R(w_i)^{R(w_i)} \pmod{p}$$

If the equality is not accepted, set another price $w_j (\neq w_i)$ and go back to step 1. Otherwise, the auctioneer can know that w_i is the bidding price w_k of the concerned bidder.

Recall we have a discussion similar to the above when arguing the *semantic security* of public key encryptions. Suppose that we have a message space $M = \{m_1, m_2\}$ and an encryption function Enc . Here, Enc is said to be *polynomially secure*, if no passive adversary can select m_1 as the corresponding message with probability significantly greater than $\frac{1}{2}$, given the encrypted message $Enc(m_1)$ (For the definition of *polynomially secure*, see [GM84, MvOV97]).

The complexity of finding $w_i = w_k$ via the above algorithm depends on the size of the message space. If the space is sufficiently large, the secrecy against the above attack is enhanced while the efficiency of opening phase is disserved. Of course, the auctioneer can access the bidding price easily when the number of elements in the message space is small.

In the next section, we present an improved protocol to enhance the security of proposed scheme up to the polynomially secure one but then it is heuristic.

5.2 An Improved Scheme

We improve the original scheme described in the section 4 against the *one-by-one attack*. Firstly, each parameter in the step.2 of bidding protocol is changed as follows.

Original Protocol		Improved Protocol
$\tilde{r} = r^x \pmod{p}$	$\longrightarrow$	$\tilde{r} = (r^r)^x \pmod{p}$
$\sigma = k - cS_A \pmod{q}$	$\longrightarrow$	$\sigma = rk - cS_A \pmod{q}$
$t = \frac{\tilde{r}\mu + rk}{S_G + \mathcal{H}(w_k, \tilde{r}, \lambda)} \pmod{q}$	$\longrightarrow$	$t = \frac{\tilde{r}\mu + rk}{S_G + \mathcal{H}(w_k, \tilde{r}, r, \lambda)} \pmod{q}$

Secondly, we describe the modified process of step.3 in the opening protocol below.

Table 2. Knowledge known to the auctioneer and public data in the improved scheme

	Bidding stage	Repudiation	Opening stage
To auctioneer	$P_G, (P_A, Cert_A, h, \tilde{r}, \lambda, \tilde{\lambda}, \sigma, t)$	—	r, x
On a bulletin board	$P_G, (h, \tilde{r}, \sigma, t)$	β	λ, r

[3a']: The bidder A sends (x, r) to the auctioneer.

[3b']: The auctioneer verifies the validity of the (individual) signature by the following formula:

$$\begin{aligned}\tilde{r} &= (r^r)^x \pmod{p} \\ \tilde{r} &= (\alpha^\sigma P_A^{\mathcal{H}(w_k, \tilde{r}, \lambda, \tilde{\lambda})})^x \pmod{p}\end{aligned}$$

with the public-key P_A . Then, the auctioneer verifies the correspondence of $\tilde{\lambda} = \lambda^x \pmod{p}$ and checks the validity of the group signature:

$$\tilde{\lambda} \tilde{r} \tilde{r}^r = (P_G \alpha^{\mathcal{H}(w_k, \tilde{r}, r, \lambda)})^{tx} \pmod{p}$$

[3c']: If both the individual signature and the group signature are correct, the auctioneer publishes (r, λ) .

[3d']: All bidders can verify the validity of the winning price with published data $(\lambda, r, \tilde{r}, t, w_k, P_G)$.

$$\lambda^{\tilde{r}} \tilde{r}^r = (P_G \alpha^{\mathcal{H}(w_k, \tilde{r}, r, \lambda)})^t \pmod{p}$$

6 Discussion

6.1 Correctness

Our scheme satisfies all requirements described in 3.3.

Secrecy of bidding price:

All bidding price of losers are never revealed to anybody including the auctioneer since they have no information on a secret x .

Validity of the successful bid:

Anybody can verify the validity of determining the successful bid with the additional knowledge (λ, r) published on the bulletin board at *Opening protocol*.

Fairness:

Nobody can tender with an advantageous condition than the other bidders (even via bid rigging and collusion with the auctioneer) since the content of bid is only known to the concerned bidder.

Anonymity of the winner:

Due to our new convertible undeniable group signature, knowledge related to the winner is never revealed to anyone but the auctioneer, while the other losers can verify both the validity of the successful bid and the fact that the winner belongs to the advanced registered group.

With respect to the anonymity of winner, we discuss another point:

Information related to the identity of each bidder is never opened in our scheme.

It is not desirable that the specific information with each bidder, which is used for the verification of the validity of bidding such as their public keys, is published on a bulletin board. Suppose that you just notice the existence of person who often becomes the successful bidder with the expensive price. It is enough to dampen your incentive to participate in the auction. Here, introducing the idea of pseudonym is no sense as a solution to this problem, since it can represent the personality of owner for an auction, but the name/identity of owner.

In our scheme, the specific information on each bidders is never opened to preclude the control of market by the abuse of publishing it.

6.2 Security against One-by-One Attack

We discuss security of the improved scheme against *one-by-one attack*.

Firstly, the auctioneer who wants to know the bidding price of bidder A via the attack computes $R(w_i)$ as follows.

$$\begin{aligned} R(w_i) &= \alpha^\sigma P_A^{c(w_i)} \pmod{p} \\ &= r^r P_A^{-c(w_k)} P_A^{c(w_i)} \pmod{p} \end{aligned}$$

Here, $R(w_i) = r^r$ if w_i is the bidding price w_k . Then, the improved scheme is still vulnerable to the *one-by-one attack*, if the auctioneer can check the correspondence of the following verification formula:

$$(P_G \alpha^{\mathcal{H}(w_k, \tilde{r}, r, \lambda)})^t \stackrel{?}{=} \lambda^{\tilde{r}} R(w_i) \pmod{p}$$

As an input into the hash function $\mathcal{H}$, r is required for the above formula. However, it is computationally infeasible that the auctioneer can compute r with no knowledge of x , given $\tilde{r} = (r^r)^x \pmod{p}$.

We analyze this problem from another viewpoint.

*Algorithm X can output r in expected polynomial time,
given a prime p and $\tilde{r} = r^r \pmod{p}$.*

If there exists the algorithm X , the auctioneer might be able to access w_k by checking the correspondence of the verification formula with candidates computed from $R(w_i)$ via the algorithm X .

It is considerable that the improved scheme is computationally *secure* against the *one-by-one attack* since there exists no such algorithm as far as we know.

7 Concluding Remarks

We have developed a new variant of convertible undeniable group signatures for achieving stronger anonymity of the winner in anonymous electronic bidding protocols.

However, the security of proposed signature scheme is heuristic and no formal argument on the security is given. Then, the next step of our research is to design such a provably secure scheme. Also we shall consider separability of group signature schemes remarked by Camenisch and Michels [CM99].

References

- BCD90. J. Boyar, D. Chaum, and I. Damgard, "*Convertible undeniable signatures*," Advances in Cryptology – CRYPTO '90, LNCS 537, pp. 189-205, 1990.
- BT97a. J. Benaloh and D. Tuinstra. "Receipt-free secret-ballot elections," Proc. STOC '94, pages 544–553.
- Cha81. D. Chaum, "*Untraceable electronic mail, return addresses, and digital pseudonyms*," Communications of ACM, Vol. 24, No. 2, pp. 84-88, 1981.
- Cha90. D. Chaum, "*Zero-knowledge undeniable signatures*," Advances in Cryptology – EUROCRYPT '90, LNCS 473, pp. 458-464, 1990.
- CvA89. D. Chaum and H. van Antwerpen, "*Undeniable Signatures*," Advances in Cryptology – CRYPTO '89, LNCS 435, pp. 212-216, 1989.
- CFSY96. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung, "*Multi-authority secret-ballot elections with linear work*," Advances in Cryptology – EUROCRYPT '96, pp.72-83, 1996.
- CvH91. D. Chaum and E. van Heyst, "*Group signatures*," Advances in Cryptology – Eurocrypt'91, LNCS 547.
- CM99. J. Camenisch and M. Michels, "*Separability and efficiency for generic group signature schemes*," Proc. CRYPTO'99, LNCS 1666.
- Dam96. I. Damgard, "*New convertible undeniable signature schemes*," Advances in Cryptology – EUROCRYPT '96, LNCS 1070, pp. 372-386, 1996.
- ebay. eBay Incorporation. <http://pages.ebay.com/>
- FIPS95. FIPS 180-1, "*Secure hash standard*," Federal Information Processing Standards Publication 180, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, 1993.
- FR95. M. K. Franklin and M. K. Reiter, "*Verifiable signature sharing*," Advances in Cryptology – EUROCRYPT '95, LNCS 921, pp. 50-63, 1995.
- FR96. M. K. Franklin and M. K. Reiter, "*The design and implementation of a secure auction service*," IEEE Transactions on Software Engineering, 22(5), pp. 302-312, 1996.
- GKR97. R. Gennaro, H. Krawczyk, and T. Rabin, "*RSA-based undeniable signatures*," Advances in Cryptology – CRYPTO '97, LNCS 1294, pp. 132-149, 1997.
- GM84. S. Goldwasser and S. Micali, "*Probabilistic encryption*" JCSS,28 (1984),
- HTK98. M. Harkavy, J. D. Tygar, and H. Kikuchi, "*Electronic auctions with private bids*," In Third USENIX Workshop on Electronic Commerce Proceedings, pp. 61-74, 1998.
- IM95. S. Inoue and T. Matsumoto, "*A note on anonymous electronic auction*," Technical Report of IEICE, ISEC95-5, 1995 (in Japanese).

- IMI94. Y. Imamura, T. Matsumoto, and H. Imai, "*Electronic anonymous bidding schemes*," The 1994 Symposium on Cryptography and Information Security, SCIS94-11B, 1994 (in Japanese).
- KHT98. H. Kikuchi, M. Harkavy, and J. D. Tygar, "*Multi-round anonymous auction protocols*," In Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems, pp. 62-69, 1998.
- KN98. H. Kikuchi and S. Nakanishi, "*Registration-free protocol for anonymous auction*," Proceedings of Computer Security Symposium '98, pp. 243-248, 1998 (in Japanese).
- Kob87. N. Kobitz, "*Elliptic curve cryptosystems*," Mathematics of Computation, 48, pp. 203-209, 1987.
- KPW96. S.J.Kim, S.J.Park, and D.H.Won, "*Convertible group signatures*," Proc. ASIACRYPT'96, LNCS 1163.
- Kud98. M. Kudo, "*Secure electronic sealed-bid auction protocol with public key cryptography*," IEICE Trans. Fundamentals, Vol. E81-A(1), pp. 20-27, January 1998.
- Mil85. V. S. Miller, "*Use of elliptic curves in cryptography*," Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 417-426, 1985.
- MPH96. M. Michels, H. Petersen, and P. Horster, "*Breaking and repairing a convertible undeniable signature scheme*," Proc. 3rd ACM Conference on Computer and Communications Security, pp. 148-152, 1996.
- MiSt97. M. Michels and M. Stadler, "*Efficient convertible undeniable signature schemes*," Proc. 4th Annual Workshop on Selected Areas in Cryptography, 1997.
<http://www.geocities.com/CapeCanaveral/Lab/8983/publications.htm>
- MvOV97. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, "HANDBOOK OF APPLIED CRYPTOGRAPHY," CRC Press, 1997.
- NWFK95. T. Nakanishi, H. Watanabe, T. Fujiwara, and T. Kasami, "*An anonymous bidding protocol using undeniable signature*," The 1995 Symposium on Cryptography and Information Security, SCIS95-B1.4, 1995 (in Japanese).
- NR94. Niemi and Renvall, "How to prevent buying of votes in computer elections," Proc. Asiacrypt'94.
- Oka97. T.Okamoto, "Receipt-free elections voting schemes for large scale elections," Proc. Security Protocol'97.
- Pet97. H. Petersen, "*How to convert any digital signature scheme into a group signature scheme*," Security Protocols, LNCS 1361, pp. 178-190, 1997.
- RD97. M. K. Reiter and A. D. Rubin, "*Crowds:anonymity for web transactions*," The ACM Transactions on Information and System Security, Vol. 1, Number 1, November 1998.
- Riv92. R. L. Rivest, "*The MD5 message-digest algorithm*," Internet Request for Comments 1321, April 1992.
- Sak99. K. Sako, "*An auction protocol which hides bids of losers*," Proc. Public Key Cryptography, LNCS 1751 (2000): a previous version is "*Universally verifiable auction protocol which hides losing bids*," Proceedings of the 1999 Symposium on Cryptography and Information Security, pp. 35-39, 1999 (in Japanese).
- SK95. Sako and Kilian, "Receipt-Free Mix-type voting scheme," Proc. Eurocrypt'95.
- SKFN98. K. Seo, H. Kikuchi, A. Fujioka, and S. Nakanishi, "*Evaluation of anonymous channel in the internet*," The 1998 Symposium on Cryptography and Information Security, SCIS98-3.3.E, 1998 (in Japanese).
- SGR97. P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "*Anonymous connections and Onion routing*," IEEE Symposium on Security and Privacy, pp. 44-54, 1997.

- SM99. K. Sakurai and S. Miyazaki, “*A bulletin-board based digital auction scheme with bidding down strategy*,” Proc. of 1999 International Workshop on Cryptographic Techniques and E-Commerce, M.Blum and C.H.Lee Eds., pp.180-187.
- SS99. S. G. Stubblebine and P. F. Syverson, “*Fair on-line auctions without special trusted parties*,” Proceedings of Financial Cryptography, 1999, LNCS 1648.
- Yah. Auctions of YAHOO, <http://auctions.yahoo.com/>.
- Yam88. S. Yamamura, “*A bidding protocol on a network*,” The Proceedings of the 1988 Workshop on Cryptography and Information Security, pp. 41-50, 1988 (in Japanese).

A Proving the Equality/Inequality of Two Discrete Logarithms [MiSt97]

We describe here the Mitchels-Stadler’s protocol of proving the equality/inequality of two discrete logarithms.

Now, the prover has a secret x satisfying that $z = \beta^x \pmod{p}$ and $y = \alpha^x \pmod{p}$, where α and β are generators of a subgroup of order q . In this paper, proving the *equality* ($\log_\beta z = \log_\alpha y$) is used as the confirmation of bidding, and the *inequality* ($\log_\beta z \neq \log_\alpha y$) is as showing the validity of negating the bid.

- Step.1:** The verifier chooses $u, v \in Z_q^*$ randomly and sends $a = \alpha^u y^v \pmod{p}$ to the prover.
- Step.2:** The prover generates random numbers $k, \tilde{k}, w \in Z_q^*$ and computes $r_\alpha = \alpha^k \pmod{p}$, $r_\beta = \beta^k \pmod{p}$, $\tilde{r}_\alpha = \alpha^{\tilde{k}} \pmod{p}$, $\tilde{r}_\beta = \beta^{\tilde{k}} \pmod{p}$. Then, the prover sends $(r_\alpha, r_\beta, \tilde{r}_\alpha, \tilde{r}_\beta, w)$ to the verifier.
- Step.3:** The verifier sends (u, v) to the prover.
- Step.4:** The prover sends $(s, \tilde{s})$ to the verifier only and only if $a = \alpha^u y^v \pmod{p}$.

$$\begin{aligned} s &= k - (v + w)x \pmod{q} \\ \tilde{s} &= \tilde{k} - (v + w)\tilde{x} \pmod{q} \end{aligned}$$

- Step.5:** The verifier checks whether:

$$\begin{aligned} \alpha^s y^{v+w} &= r_\alpha \pmod{p} \\ \alpha^{\tilde{s}} r_\alpha^{v+w} &= \tilde{r}_\alpha \pmod{p} \\ \beta^{\tilde{s}} r_\beta^{v+w} &= \tilde{r}_\beta \pmod{p} \end{aligned}$$

and finally verifies:

$$\begin{cases} (\text{Confirmation}) : \beta^s z^{v+w} = r_\beta \pmod{p} \\ (\text{Repudiation}) : \beta^s z^{v+w} \neq r_\beta \pmod{p} \end{cases}$$

AKA Protocols for Mobile Communications^{*}

KookHeui Lee¹ and SangJae Moon²

¹ Telecommunication R&D Center, Samsung Electronics Co. LTD.,
Samsung Plaza Bldg., Sungnam, Korea
`kookheui@telecom.samsung.co.kr`

² School of Electronic and Electrical Eng., Kyungpook National University,
Taegu, 702-701, Korea
`sjmoon@knu.ac.kr`

Abstract. This paper proposes a modified signcryption scheme that provides signer anonymity. In addition, an efficient 2-pass authentication and key agreement protocol that uses the proposed signcryption scheme is presented for the mobile communication air-interface. The use of the modified scheme in the implementation of the ASPeCT protocol is demonstrated to generate a lower computational load than other methods. A modified ASPeCT protocol, which satisfies forward secrecy, is also proposed.

1 Introduction

Public key cryptography is being utilized in the implementation of security services for future mobile communications due to its key management facility and variety of security services. When designing an authentication and key agreement (hereinafter AKA) protocol, certain restrictions caused by mobile communication environments must be taken into consideration. One of these is that a mobile user device has a limited computational capability compared to fixed networks. Accordingly, a smaller computational load for the mobile user device is desirable. Moreover, the bandwidth between a mobile user and a fixed network is generally expensive and limited, therefore, the length of the exchanged messages needs to be kept as short as possible.

Several AKA protocols [1,2,3,4] for mobile communications currently employ the MSR (modular square root) cryptosystem [5] for a smaller computational load on the mobile side since the MSR cryptosystem only requires one modular multiplication by the mobile user device. However, the MSR cryptosystem can not be converted into an efficient ECC (elliptic curve cryptography) system [6, 7,8], which is a disadvantage from the perspective of the bandwidth.

Other AKA protocols [9,10] use off-line computation to achieve prompt AKA protocol runs. These protocols adopt a cryptosystem based on a DLP (discrete logarithm problem) that can be easily converted to an ECDLP (elliptic curve discrete logarithm problem). Even though these protocols involve more intensive

^{*} This work was supported by the Brain Korea 21 Project.

computations than a MSR cryptosystem, a mobile user can compute the most time consuming exponentiations prior to initiating an AKA protocol. However, despite such endeavors, the total computational load at the mobile user device can not be reduced in two particular cases; signature generation and public key encryption, or signature generation and Diffie-Hellman key exchange.

A signcryption scheme [11,12,13,14] is a cryptographic method that involves the functions of encryption and digital signature simultaneously. Such a scheme has fewer computations than those required by traditional signature-then-encryption and assumes that the signer and verifier have prior knowledge of the other's public key. As a result, this type of scheme can not be employed with an AKA protocol using radio channels because user identity information, such as an identifier, public key, or user certificate, can not be sent over an air-interface due to the security requirement of user anonymity for AKA protocols in mobile communications [15].

This paper proposes a modified signcryption scheme suitable for mobile communications. The proposed signcryption scheme requires a slightly higher computational load than a conventional signcryption scheme [11], yet fewer computations than a conventional signature-then-encryption scheme. In addition, an efficient 2-pass AKA protocol which uses the concept of signature-then-encryption is presented for a mobile communication air-interface. It is also demonstrated that use of the proposed signcryption scheme in the implementation of the AS-PeCT protocol produces a lower computational load than other methods. Plus, a modified ASPeCT protocol using the proposed signcryption scheme is presented which provides forward secrecy with only slightly more computation, however, the number of exponentiations on the mobile side remains unchanged.

2 Proposal of Modified Signcryption Scheme

The following notations are used to describe the protocols throughout this paper.

p :	large prime number
q :	large prime factor of $p - 1$
g :	element of $\mathbf{Z}_p^*$ of order q
ID_E :	identifier of entity E
$RT(TS)$:	real time value (time stamp)
$hash$:	one-way hash function
KH :	keyed one-way hash function
x_E :	secret key of entity E
P_E :	public key of entity E
K_{MB} :	common session key between M and B
$Cert_E$:	certificate of P_E
R_E :	random number generated by entity E
$E_K\{x\}(D_K\{x\})$:	symmetric encryption (decryption) of x using key K
$x \parallel y$:	concatenation of x and y

2.1 Description of Proposal

Fig. 1 describes the modified signcryption scheme. It is assumed that Alice has message $\mathbf{m}$ to send to Bob. Alice signcrypts it so that the effect is similar to signature-then-encryption.

Alice [Signcryption]	Bob [Unsigncryption]
$\mathbf{K} = \text{hash}(\mathbf{P}_B^{r_A} \bmod p)$ $\mathbf{T} = \mathbf{g}^{r_A} \bmod p$ $\mathbf{r} = \mathbf{KH}_T(\mathbf{m}) \quad \Rightarrow \mathbf{T} \parallel \mathbf{RT} \parallel \mathbf{c} \Rightarrow$ $\mathbf{s} = r_A / (\mathbf{r} + \mathbf{x}_A) \bmod q$ $\mathbf{c} = \mathbf{E}_K\{\mathbf{m} \parallel \mathbf{r} \parallel \mathbf{s}\}$	$\mathbf{K} = \text{hash}(\mathbf{T}^{x_B} \bmod p)$ $\mathbf{D}_K\{\mathbf{c}\} = \mathbf{m} \parallel \mathbf{r} \parallel \mathbf{s}$ $\mathbf{T} ? = (\mathbf{P}_A \cdot \mathbf{g}^r)^s \bmod p$ $\mathbf{KH}_T(\mathbf{m}) ? = \mathbf{r}$

Fig. 1. Modified signcryption scheme

[Signcryption]

1. Randomly choose $r_A \in_R \mathbf{Z}_q^*$
2. Compute encryption key $\mathbf{K} = \text{hash}(\mathbf{P}_B^{r_A} \bmod p)$ and $\mathbf{T} = \mathbf{g}^{r_A} \bmod p$
3. Compute $\mathbf{r} = \mathbf{KH}_T(\mathbf{m})$, and $\mathbf{s} = r_A / (\mathbf{r} + \mathbf{x}_A) \bmod q$ which serve as signature
4. Encrypt $\mathbf{r}$, $\mathbf{s}$, and some other data using $\mathbf{K}$
5. Send $\mathbf{T}$ to Bob together with $\mathbf{RT} \parallel \mathbf{c}$

[Unsigncryption]

1. Bob takes $\mathbf{T}$ from the received message and computes the decryption key $\mathbf{K} = \text{hash}(\mathbf{T}^{x_B} \bmod p)$
2. Decrypt the received message $\mathbf{c}$ using $\mathbf{K}$ to get $\mathbf{m}$, $\mathbf{r}$, and $\mathbf{s}$
3. Compute $(\mathbf{P}_A \cdot \mathbf{g}^r)^s \bmod p$ and compare with received $\mathbf{T}$
4. Compute $\mathbf{KH}_T(\mathbf{m})$ and compare with decrypted $\mathbf{r}$

If necessary, Bob can then forward $(\mathbf{m}, \mathbf{r}, \mathbf{s})$ to others, who will be convinced that it came originally from Alice by verifying

$$\mathbf{K} = (\mathbf{P}_A \cdot \mathbf{g}^r)^s \bmod p \quad \text{and} \quad \mathbf{r} = \mathbf{KH}_K(\mathbf{m}) \quad (1)$$

2.2 Security Features and Computational Load

A new type of cryptographic primitive called signcryption was introduced in 1997 by Zheng [11], which includes simultaneous message encryption and digital signature. It involves fewer computations and has a lower communication overhead than conventional signature-then-encryption approaches. In Zheng's

scheme, the unsigncryption (decryption and signature verification) requires the recipient's private key, which can produce certain constraints in applications where signatures need to be validated by others.

Bao and Deng [12] proposed a directly-verifiable signcryption scheme that has a higher computational load than Zheng's scheme and yet can prevent a key recovery attack by a judge [16]. Bao and Deng's signcryption scheme has been subsequently modified for specific applications [13,14]. However, despite its low communication overhead, small computational load, and suitability for mobile communications, this scheme can not be directly applied to mobile communications because it does not satisfy the security requirement of signer anonymity.

The first step of the unsigncryption procedure of the conventional signcryption schemes in [11,12,13,14] includes the public key of the signer as follows.

$$\mathbf{K} = \text{hash}(\mathbf{P}_A \cdot \mathbf{g}^{\mathbf{x}})^{s \cdot \mathbf{x}_B} \bmod \mathbf{p} \quad (2)$$

This means that the recipient can compute the decryption key if the public key of the signer is received in advance. However, in mobile communications the public key of the signer(the mobile user) needs to be encrypted to satisfy signer anonymity.

In the proposed scheme in Fig. 1, the recipient can only generate the decryption key with the received data ($\mathbf{T}$) and his own secret key ($\mathbf{x}_B$) such that

$$\mathbf{K} = \text{hash}(\mathbf{T}^{\mathbf{x}_B} \bmod \mathbf{p}) \quad (3)$$

As a result, the public key of the signer can be encrypted using the key. Therefore, the proposed scheme can ensure the anonymity of the signer(mobile user).

The number of modular exponentiations for the proposed signcryption scheme can be calculated based on Fig. 1. Table 1 shows a comparison of the computational loads of the various methods. The proposed scheme involves the same number of computations as the other schemes except for Zheng's scheme, however, this scheme has a weakness related to key recovery attack [16].

Table 1. Comparison of number of exponentiations

Operation	Proposed scheme	Zheng's scheme	Bao and Deng's scheme	DSA signature + ElGamal encryption
Signcryption	2	1	2	1 + 2
Unsigncryption	3	2	3	1 + 2

3 2-Pass AKA Protocol

3.1 Description of Protocol

The concept of signcryption was adopted to reduce the computational load in a 2-pass AKA protocol [17]. Fig. 2 presents a flow diagram of the protocol in which the anonymity of the mobile user is ensured. Its procedure is executed as follows.

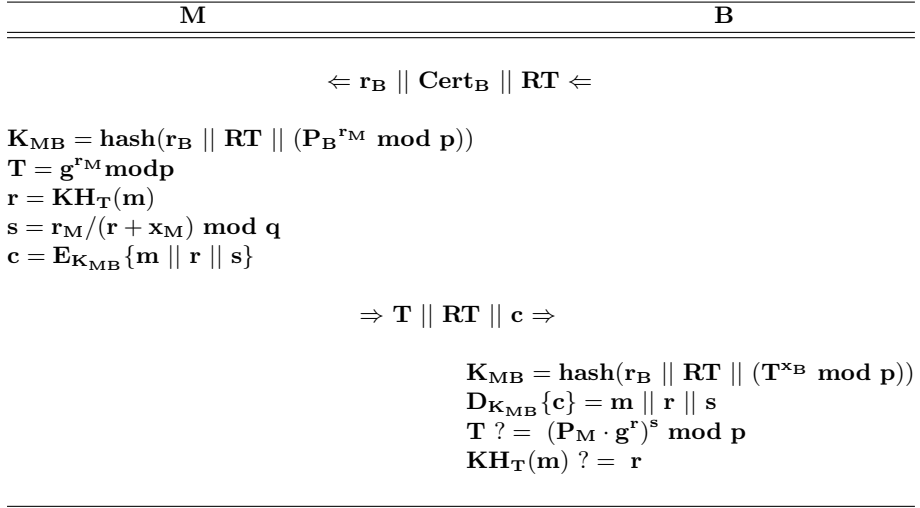


Fig. 2. Flow diagram of 2-pass AKA protocol

First pass from *B* to *M*

- **B** broadcasts a random number $\mathbf{r}_B$, real-time value $\mathbf{RT}$, and its public key certificate $\mathbf{Cert}_B$, where the identifier $\mathbf{ID}_B$ and **B**'s public key $\mathbf{P}_B$ are included in the certificate $\mathbf{Cert}_B$.

Second pass from *M* to *B*

- Entity **M** executes the following actions :
 1. Extract $\mathbf{ID}_B$ and $\mathbf{P}_B$ from $\mathbf{Cert}_B$
 2. Generate random number $\mathbf{r}_M$
 3. Compute common session key $\mathbf{K}_{MB} = \text{hash}(\mathbf{r}_B \parallel \mathbf{RT} \parallel (\mathbf{P}_B^{\mathbf{r}_M} \bmod \mathbf{p}))$ and temporary key $\mathbf{T} = \mathbf{g}^{\mathbf{r}_M} \bmod \mathbf{p}$
 4. Compute $\mathbf{r}$ and $\mathbf{s}$ which serve as signature
 5. Encrypt $\mathbf{r}$, $\mathbf{s}$, and other data $\mathbf{m}$ including $\mathbf{Cert}_M$ by using $\mathbf{K}_{MB}$
 6. Send $\mathbf{T}$ and $\mathbf{RT}$ to **B** together with encrypted message

- Entity **B** executes the following actions :
 1. **B** takes **T** and **RT** from received message and computes common session key $K_{MB} = \text{hash}(T^{x_B} \bmod p)$
 2. Decrypt received message **c** using K_{MB} and extract **M**'s identifier and public key from Cert_M
 3. Compute $(P_M \cdot g^r)^s \bmod p$ and compare with received **T**. Protocol is aborted if received **T** is invalid
 4. Compute $KH_T(m)$ and compare with decrypted **r**

3.2 Security Features

Based on the security requirements in [15], the security features are as follows.

- Entity authentication: Explicit authentication of **M** to **B**. Since **M** signcrypts on a random number r_B generated by **B**, **B** can explicitly authenticate **M**.
- Key authentication: Explicit key authentication to **B** and implicit key authentication to **M**. Since **M** actually creates the key **K** and then signcrypts using it, the verifier **B** can have the assurance that **K** is actually computed by the signer **M**. Since the key **K** can be only computed by a holder of the secret x_B , the sender **M** can have an implicit key authentication.
- Key agreement: Mutual agreement of session key between **M** and **B**. This is because the key is derived by the randomly chosen numbers r_M and r_B by **M** and **B**, respectively.
- Key confirmation: Explicit key confirmation to **B** and implicit key confirmation to **M**. The former property is based on the fact that **M** signcrypts based on the generated session key **K**. Whereas the latter is based on the fact that **M** is able to believe that the encrypted message can only be decrypted by a receiver who knows the secret key x_B .
- Key freshness: Mutual assurance of key freshness. Ensuring the freshness of the generated random number ensures the freshness of the session key.
- Anonymity of mobile user: Since any information on **M** is encrypted in the protocol, anonymity is ensured.
- Non-repudiation of user: This property is based on the usage of the signature in the signcrypton scheme.

4 Implementation of ASPeCT Protocol Using Proposed Signcrypton

The ASPeCT protocol [15] is basically designed to shift as much computational load as possible from the mobile user terminal to a base station, because it is assumed that the mobile user will be represented by a smart card which has a limited computational capability. With the use of the proposed signcrypton, the ASPeCT protocol can be implemented as an efficient 3-pass AKA protocol.

4.1 Description of Protocol

Fig. 3 describes the flow of the ASPeCT protocol when the proposed signcryption is employed, and its procedures can be illustrated as follows.

First pass from M to B

- When a mobile user wants to initiate a communication session, they send $\mathbf{T} = \mathbf{g}^{r_M} \bmod p$ to a base station. Upon receiving the first message from the mobile user, the base station generates a random number $\mathbf{r}_B$ and calculates the common session key $\mathbf{K}_{MB} = \text{hash}(\mathbf{r}_B \parallel (\mathbf{T}^{x_B} \bmod p))$.

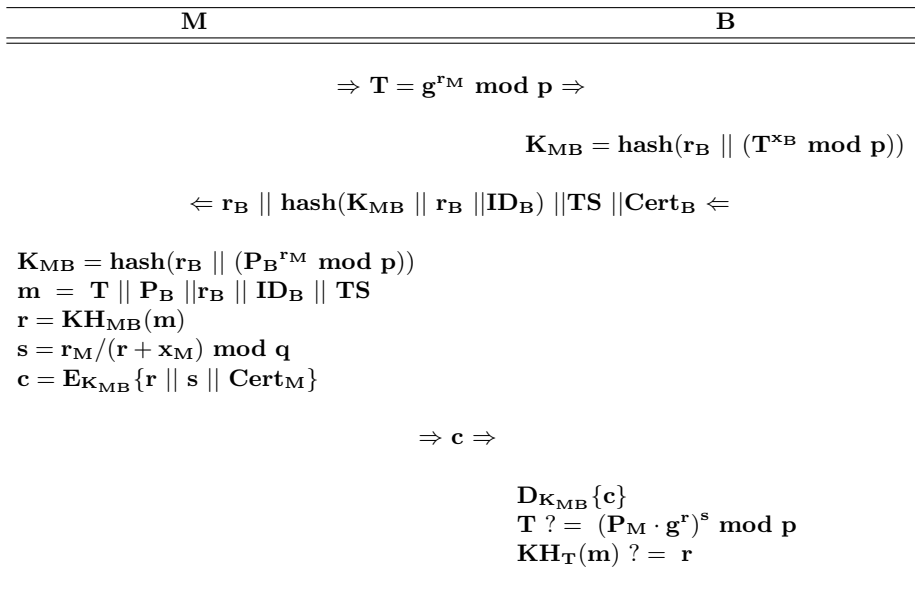


Fig. 3. Implementation of ASPeCT protocol using proposed signcryption scheme

Second pass from B to M

- In the second message, the base station sends its certificate $\mathbf{Cert}_B$, random value $\mathbf{r}_B$, and timestamp $\mathbf{TS}$ and signifies knowledge of the common session key $\mathbf{K}_{MB}$ by sending a copy of it hashed with the random numbers $\mathbf{r}_B$ and $\mathbf{ID}_B$.
- Entity M executes the following actions :
 1. Verify authenticity of base station's public key $\mathbf{P}_B$
 2. Calculate common session key $\mathbf{K}_{MB} = \text{hash}(\mathbf{r}_B \parallel (\mathbf{P}_B^{r_M} \bmod p))$

3. Verify authenticity of common session key by comparing received $\text{hash}(\mathbf{K}_{\text{MB}} \parallel \mathbf{r}_{\text{B}} \parallel \mathbf{ID}_{\text{B}})$ with own computed hash value.
4. Generate signatures $\mathbf{r}$ and $\mathbf{s}$
5. Encrypt signature with own certificate using established common session key $\mathbf{K}_{\text{MB}}$

Third message from M to B

- On receipt of the third message, B executes the following actions :
 1. Decrypt received message using common session key
 2. Compute $(\mathbf{P}_{\text{M}} \cdot \mathbf{g}^{\mathbf{r}})^{\mathbf{s}} \bmod \mathbf{p}$ and compare with received $\mathbf{T}$. Protocol is aborted if received $\mathbf{T}$ is invalid
 3. Compute $\mathbf{KH}_{\mathbf{K}_{\text{MB}}}(\mathbf{m})$ and compare with decrypted $\mathbf{r}$

4.2 Computational Load

When designing the ASPeCT protocol, a key agreement scheme (similar to the ElGamal scheme [18]) with an implicit key authentication [19] of the base station was used because the entity authentication of the base station can be obtained without much extra computational load. The common session key for the AS-PeCT protocol is given by

$$\mathbf{K}_{\text{MB}} = \text{hash}(\mathbf{r}_{\text{B}} \parallel (\mathbf{P}_{\text{B}}^{\mathbf{r}_{\text{M}}} \bmod \mathbf{p})) = \text{hash}(\mathbf{r}_{\text{B}} \parallel ((\mathbf{g}^{\mathbf{r}_{\text{M}}})^{\mathbf{r}_{\text{B}}} \bmod \mathbf{p})) \quad (4)$$

In the proposed signcryption scheme, both participants also establish the encryption and decryption keys using the ElGamal scheme.

$$\mathbf{K} = \text{hash}(\mathbf{P}_{\text{B}}^{\mathbf{r}_{\text{A}}} \bmod \mathbf{p}) = \text{hash}((\mathbf{g}^{\mathbf{r}_{\text{M}}})^{\mathbf{x}_{\text{B}}} \bmod \mathbf{p}) \quad (5)$$

Accordingly, the modified 3-pass AKA protocol reduces the computational load of the mobile user through the use of a common session key as the encryption key for the modified signcryption scheme. Table 2 confirms that using the modified signcryption scheme can reduce the computational load of the ASPeCT protocol.

Table 2. Comparison of number of exponentiations

Entity	Modified protocol	ASPeCT protocol
Mobile user	2	3
Base station	3	3

5 3-Pass AKA Protocol with Forward Secrecy

5.1 Forward Secrecy

A protocol will satisfy forward secrecy if the session keys established by long-term secret keys in a protocol run are not compromised even if the long-term keys are disclosed. Recently, Park etc. [20] has treated this concept for application in mobile communications. Accordingly, they presented two modified ASPeCT protocols that provide forward secrecy. Generally, forward secrecy requires additional computations in a protocol run. Therefore, a protocol that can include forward secrecy with a small computational load is highly desirable.

The ASPeCT protocol establishes a common session key $\mathbf{K} = \text{hash}(\mathbf{r}_B \parallel (\mathbf{g}^{\mathbf{r}_M \cdot \mathbf{x}_B} \bmod \mathbf{p}))$ using a variant of the ElGamal scheme. As a result, the ASPeCT protocol avoids certain public key based computations and thereby succeeds in turning on-line exponentiation within the mobile user into off-line exponentiation because, in most cases, the mobile user can take the advantage of the pre-knowledge of the base station's public key $\mathbf{P}_B$. However, this variant of the ElGamal scheme does not assure forward secrecy as, if the long-term secret key of the base station is compromised and all the protocol transcripts for a particular session are recorded (for the knowledge of $\mathbf{g}^{\mathbf{r}_M}$ and $\mathbf{r}_B$) by an attacker, the common session key for the session can be easily deciphered by the attacker. However, the disclosure of the secret key of the mobile user alone does not lead to the disclosure of the common session key (so partial forward secrecy is satisfied), and in this case, the real problem is rather one of authentication than forward secrecy. Moreover, forward secrecy concerns a mobile user who cannot ensure that the secret key of the base station will not be compromised.

5.2 Description of Protocol

Fig. 4 illustrates the flow of the modified ASPeCT protocol, which satisfies forward secrecy.

First pass from M to B

- When M wants to initiate a communication session, the send $\mathbf{T} = \mathbf{g}^{\mathbf{r}_M} \bmod \mathbf{p}$ to B
- Upon receiving the first message B executes the following actions:
 1. Generate random number $\mathbf{r}_B$ and calculate $\mathbf{g}^{\mathbf{r}_B} \bmod \mathbf{p}$
 2. Calculate common session key $\mathbf{K}_{MB} = \text{hash}(\mathbf{T}^{\mathbf{r}_B} \bmod \mathbf{p})$ and authentication response value $\mathbf{Auth} = \text{hash}(\mathbf{T}^{\mathbf{x}_B} \bmod \mathbf{p})$

Second pass from B to M

- In the second message, B sends their certificate $\mathbf{Cert}_B$, computed $\mathbf{g}^{\mathbf{r}_B} \bmod \mathbf{p}$, timestamp $\mathbf{TS}$, and authentication response value $\mathbf{Auth}$ and signifies knowledge of the common session key $\mathbf{K}_{MB}$ by sending a copy of it hashed with the random numbers $\mathbf{r}_B$ and $\mathbf{ID}_B$.

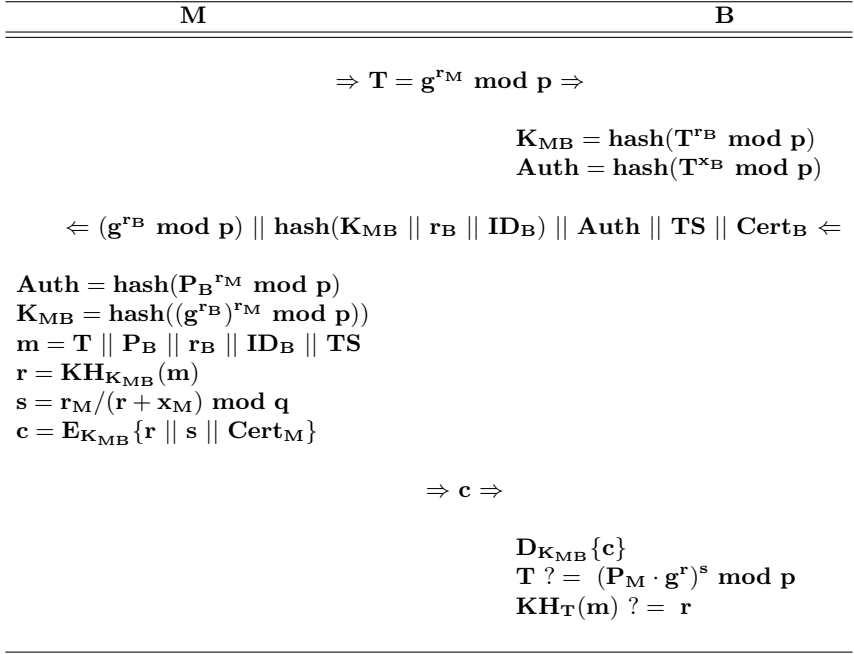


Fig. 4. 3-pass AKA protocol with forward secrecy

- Entity **M** executes the following actions:
 1. Verify authenticity of **B** by computing and comparing **Auth**
 2. Verify authenticity of **B**'s public key **P_B**
 3. Calculate common session key $K_{MB} = \text{hash}((g^{r_B})^{r_M} \bmod p)$
 4. Verify authenticity of common session key by comparing received $\text{hash}(K_{MB} \parallel r_B \parallel \text{ID}_B)$ with own computed value.
 5. Generate signatures **r** and **s**
 6. Encrypt signature with own certificate using established common session key **K_{MB}**

Third message from *M* to *B*

- On receipt of the third message, **B** executes the following actions :
 1. Decrypt received message using common session key
 2. Compute $(P_M \cdot g^r)^s \bmod p$ and compare with received **T**. Protocol is aborted if received **T** is invalid
 3. Compute $\text{KH}_{K_{MB}}(m)$ and compare with decrypted **r**

5.3 Computational Load

In the proposed 3-pass AKA protocol, the mobile user and base station both establish a common session key $\mathbf{K} = \text{hash}(\mathbf{g}^{\mathbf{r}_M \cdot \mathbf{r}_B} \bmod \mathbf{p})$ using the Diffie-Hellman key exchange scheme based on a temporary secret key for the entities $\mathbf{r}_M$ and $\mathbf{r}_B$.

Table 3. Comparison of number of exponentiations

Entity	Modified protocol	ASPeCT protocol
Mobile user	[2] + 1	[3]
Base station	[1] + 4	3

[] : corresponding operation can be performed in off-line state

Therefore, even if the secret key $\mathbf{x}_M$ or $\mathbf{x}_B$ is disclosed to an adversary, the established common session key will not be compromised. Table 3 evaluates the computational load of the proposed 3-pass AKA protocol. The modified protocol ensures perfect forward secrecy with only a slightly heavier computational load than the ASPeCT protocol.

6 Conclusion

A signcryption scheme was modified to provide signer anonymity that it is applicable to mobile communications. The main idea is that both communicators initially establish an encryption key to protect the information data of the sender, thereafter the recipient generates a decryption key using the decrypted data and his own secret key. The modified signcryption scheme has the same computational load as that of other signcryption methods.

Employing the proposed scheme, an efficient and secure 2-pass AKA protocol was presented for the air-interface of mobile communications, along with two 3-pass AKA protocols, related to the ASPeCT protocol. The first 3-pass AKA protocol reduced the computational load on the mobile user end by one modular exponentiation without any reduction of security, whereas the other provided forward secrecy with a slightly higher computation, where the number of exponentiations on mobile side remained unchanged.

References

1. M. J. Beller, L.-F. Chang, Y. Yacobi: Privacy and authentication on a portable communications system, IEEE Journal on Selected Areas in Communications, vol. 11, pp. 821–829, Aug. 1993.

2. M. J. Beller, Y. Yacobi: Fully-fledged two-way public key authentication and key agreement for low-cost terminals, *Electronics Letters*, vol. 29, no. 11, pp. 999–1001, May 1993.
3. U. Carlsen: Optimal privacy and authentication on a portable communications system, *ACM Operating Systems Review*, vol. 28, no. 3, pp. 16–23, Jul. 1994.
4. JTC/J-STD-14: 1995, Personal Communication Services, PACS Air Interface Specification.
5. M. O. Rabin: Digitalized signatures and public-key functions as intractable as factorization, MIT Lab. Comp. Sci., Technical Report, Jan. 1979.
6. V. Miller: Uses of elliptic curves in cryptography, In H. C. Williams, editor, *Advances in Cryptology - Crypto '85*, LNCS 218, pp. 417–426, 1985.
7. N. Koblitz: Elliptic curve cryptosystems, *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
8. A. J. Menezes: *Elliptic Curve Public Key Cryptosystems*, Boston, MA: Kluwer Academic Publishers, 1993.
9. Y. Zheng: An authentication and security protocol for mobile computing, *Proceedings of IFIP*, pp. 249–257, Sep. 1996.
10. G. Horn, B. Preneel: Authentication and payment in future mobile systems, *Compute Security - ESORICS'98*, LNCS 1485, pp. 277–293, Sep. 1998.
11. Y. Zheng: Digital signcryption or how to achieve cost (signature & encryption) $\ll$ cost (signature) + cost (encryption), *Advances in Cryptology - CRYPTO'97*, LNCS 1294, pp. 165–179, Aug. 1997.
12. F. Bao, R. H. Deng: A signcryption scheme with signature directly verifiable by public key, *Proc. of PKC'98*, LNCS 1431, pp. 55–59, 1998.
13. C. Gamage, J. Leiwo, Y. Zheng: Encrypted message authentication by firewalls, *Proc. of PKC'99*, LNCS 1560, pp. 69–81, 1999.
14. M. Seo, K. Kim: Electronic funds transfer protocol using domain-verifiable signcryption scheme, *Pre-Proc. of ICISC'99*, pp. 263–271, 1999.
15. G. Horn, K. M. Martin, C. J. Mitchell: Authentication protocols for mobile network environment value-added services, draft, available at <http://isg.rhnc.ac.uk/cjm/Chris.Mitchell.htm>.
16. H. Petersen, M. Michels: Cryptanalysis and improvement of signcryption schemes, *IEE Proc. Comput. Digit. Tech.*, vol. 145, no. 2, 1998.
17. K. Lee, S. Moon, W. Jeong, T. Kim: A 2-pass Authentication and Key Agreement Protocol for Mobile Communications, *Proceedings of The 2nd International Conference on Information Security and Cryptology, ICISC'99*, pp. 143–155, Dec. 1999.
18. T. ElGamal: A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
19. ISO/IEC/11770-3: 1997, Information Technology - Security Techniques - Key Management, Part 3: Mechanisms Using Asymmetric Techniques.
20. D. Park, C. Boyd, S. Moon: Forward Secrecy and Its Application to Future Mobile Communications Security, *PKC2000*, LNCS 1751, Springer-Verlag, 2000, pp. 433–445, Jan. 2000.

A Three Phased Schema for Sealed Bid Auction System Design^{*}

Kapali Viswanathan, Colin Boyd, and Ed Dawson

Information Security Research Centre
Queensland University of Technology
Brisbane, Australia.
{viswana, boyd, dawson}@fit.qut.edu.au

Abstract. Auction systems allow many entities, each possessing a unique secret value, to interact in order to reveal the *winning* value from the set of secret values, based on the rules of interaction. A platform to achieve this goal is *sealed bid auction*. We propose a schema and a concrete design to achieve this goal in a simple, efficient and secure fashion. The schema will facilitate the use of existing and future knowledge in providing anonymity.

Keywords: Sealed bid auction, sealing process, anonymity.

1 Introduction

The fundamental goal of auction systems is the distribution of scarce resources among, potentially, many bidders based on well devised rules to determine the winning strategy [8]. A common approach to protect the interests of individual bidders, from conspiring bidders and auctioneers, is the sealed bid auction system. A *seal* is employed to provide secrecy for a bid, until a pre-defined event. In the physical world, the sealed bid may simply be a sealed envelope that encloses a paper containing the value of the bid (along with optional non-repudiation information from the bidder). The sealing process guarantees a fair auction procedure for *honest bidders*. At the same time, there must be mechanisms to open the seal (after a specified event) to reveal the winning bidder, in order to avoid disavowal after participation. A requirement for some systems, *but not necessarily for the sealing method*, is to protect the secrecy of the losing bids, even after the completion of the auction procedure. This requirement is mainly to provide restricted privacy for the losing bidders. We stress the word restricted because once the identity of the bidders is known and the identity of the winning bidder and the corresponding bid value are published, automatically some information about the bid values of the losing bidders is revealed. The only approach to provide complete privacy for losing bidders would be to refrain from publishing the identity of all the bidders.

In order to electronically implement the sealed bid auction procedure, the first step is to design a suitable *sealing process*. Towards this end the requirements

^{*} Research supported by the Australian Research Council grant A49804059

specific to the sealing process must be identified. Once such a sealing process is devised, this abstraction can be used along with other techniques to achieve a complete auction system.

1.1 Literature Review

We shall now review selected proposals from the literature. Confidentiality of the bid has been of paramount importance for the design of electronic auction systems. To achieve confidentiality of bid some proposals [5,7] used secret-sharing primitives to distribute the value of the bid among many trustees. When at least a threshold of the trustees are honest, they will not assist in opening the bid *before* the closing period. This approach generally results in inefficient systems, when public verifiability is required. This is because there exists no efficient protocol construct for publicly verifiable encryption [14] which is an essential building block for publicly verifiable secret sharing schemes. The other approach to publicly verifiable secret sharing is that of Schoenmakers [13], which is more efficient than the scheme by Stadler [14]. However, its application to the auction scheme will remain inefficient, as compared with our scheme. We shall provide an estimation for the number of exponentiations required for this approach in Section 4.3.

Sakurai and Miyazaki [11] proposed an elegant auction system where the confidentiality of bid is controlled only by the bidder. For non-repudiation of the bid, Sakurai and Miyazaki used the undeniable signature scheme. Unfortunately, the computational and communicational complexity of their scheme [11] is dependent on the number of participants, thereby rendering their system inefficient for large scale auction systems. Moreover, their scheme requires every bidder to be on-line, which may not be a desirable property in large scale auctions over open networks.

Sako [10] attempted to modify their proposal [11] using group encryption (for a group of trusted auctioneers), instead of the undeniable signature scheme, for sealing the bid. We believe that in doing so, the proposal by Sako lost the main advantage of the proposal by Sakurai and Miyazaki, which was user-controlled confidentiality of the bid.

Harkavy *et al* [6] proposed an auction scheme based on secure distributed computing primitives. Although they claim the system to be moderately efficient, the security arguments for their scheme remain unclear.

The properties that have been identified for sealed bid auction systems are:

Confidentiality of bid: Only the bidder must know the bidding strategy until the closing period.

Non-repudiation of bid: The winning bidder must not be able to repudiate or change the bidding strategy.

Publicly verifiable auction: Any monitor must be able to verify the validity of the auction procedure.

Anonymity of bidder: The bidder-bid relationship must be known only to the bidder, unless the bid conforms with the winning strategy.

Independence of auction rules: The security protocols for auction rules must be independent of the auction rules.

1.2 The Approach

We shall present the approach taken by our paper to design an auction system. The system consists of two sub-systems, an anonymity sub-system that provides anonymity to all its users and an auction sub-system that allows the users to participate in the auction procedure. Thus the system, in effect, provides an anonymous auctions service. The auction sub-system can be explained in terms of the following physical world entities. The auction system consists of a “magic seal,” that will allow only the entity that sealed the bid, to open it. The bidders place their bid values inside an envelope and apply the “magic seal” to it. In order to register in a particular auction protocol, the bidders send the envelope to the auctioneer using (something like) the *registered post* service, which guarantees that the sealed bid will reach the auctioneer (who will not be able to repudiate the receipt). When the actual auction procedure starts the bidders assist the auctioneer to break the “magic seal.” The envelopes that are not opened are discarded and do not participate in the auction.

Organisation and Notation. The paper is organised to present an abstraction of the electronic cash technology in the next section, an abstraction and design of a sealing process in Section 3 and the auction system, which employs the e-cash and sealing techniques, in Section 4.

Entities will be represented using calligraphic symbols. The following symbols possess the following semantics:

1. $:=$ is an assignment operation used only with a process, function or a protocol abstraction.
2. $\stackrel{?}{=}$ is the checking operation.
3. $\in_R$ represents the action of choosing at random.
4. $\langle X \rangle$ proof transcript (a set of values conforming to a grammar), named X .

2 Background

Our proposal will make use of an anonymous token issuer (ATI), such as the electronic cash technology, to provide privacy service for the participants in the auction process. Towards this end, we will employ the fair electronic cash scheme proposed by Frankel *et al* [3]. Note that any anonymous token issuing facility that supports *non-transferability* and *revocable anonymity* can be employed. The remaining of this section will describe the abstractions of the ATI that will be used to explain the auction protocol.

System Settings. The bank, $\mathcal{B}$, chooses and publishes primes p and q such that $p = 2q + 1$ and the generators g and g_1 of order q . The bank publishes its

public keys $y_B = g^{X_B}$ and related system setting information, and retains the secrecy of the corresponding private key. The bank also publishes the public key of the trustee, $\mathcal{T}$, (for tracing purposes) which would be of the form $f_T = g^{X_T}$, where X_T is the corresponding private key. Every user registers with the bank to obtain an identity of the form $I = g^{u_1}$, where g is the base and u_1 is the user's private key.

Protocol Withdraw: The *user* identifies himself/herself to the *bank* (or the mint) and then engages in this protocol to obtain a restrictive blind signature on a *pseudonym* of the form $A = (Ig_1)^s$, where g_1 is a base and s a secret, random value. The restrictive blind signature, restricts the structure of A to be of this form. The value of A is never revealed to the bank. We shall express this phase as, $\langle A, Cert_A \rangle_I := Withdraw(I, \mathcal{B}, \{s\}_I, \{X_B\}_B)$, which should be read as, “ I engages in the withdraw protocol with $\mathcal{B}$ using a (random) value s (known only to I) to obtain a certificate $Cert_A$ for A , which are known only to I , signed by the bank using its private key X_B .”

Protocol Spend: The *user* derives two pseudonyms of the form $A_1 = g^{u_1 s}$ and $A_2 = g_1^s$ from A , such that $A = A_1 A_2$. The user then proves to the *merchant* its knowledge of the pre-images of A_1 and A_2 with respect to the reference bases g and g_1 respectively (thereby proving the knowledge of representation of A). Additionally, he/she proves that his/her identity, which is *hidden* in A_1 , has been encrypted for a trustee under the public key f_T resulting in the ciphertext D . The user never reveals his/her identity, I , to the merchant. We shall express this phase as, $\langle Proof_A \rangle := Spend(A, Cert_A, \mathcal{M}, y_b, f_T, \{s, u_1\}_A)$, which should be read as, “ A engages in the spend protocol with $\mathcal{M}$ using the certificate, $Cert_A$, $\mathcal{B}$'s public key y_B and the private data (s, u_1) to generate the transcripts for a proof system $\langle Proof_A \rangle$, which contains an encryption of the identity of the user under the public key f_T .” $\langle Proof_A \rangle$ contains the following tuples, $(A_1, A_2, Encryption_{f_T}(I))$ along with the corresponding proof transcripts. Here $Encryption_{f_T}(I)$ is the encryption of the user identity I under the public key f_T .

Protocol Deposit: The *merchant* submits the proofs of knowledge, which it received in the Protocol Spend, to the *bank* and avails credit. The bank can check if it has already received the tuple (A_1, A_2) to detect double spent transcripts. We shall express this phase as, $Deposit(\mathcal{M}, \mathcal{B}, A, Cert_A, \langle Proof_A \rangle)$, which should be read as, “ $\mathcal{M}$ engages in the deposit protocol with $\mathcal{B}$ to submit the values $\langle A, Cert_A \rangle$ and $\langle Proof_A \rangle$.”

Protocol Trace: The *bank*, or any other authorised entity, needs to trace the identity of the user who spent a particular transcript, it can contact the *trustee*. When the trustee is provided with the transcript $\langle Proof_A \rangle$, it can retrieve the ciphertext $Encryption_{f_T}(I)$ and decrypt it using its private key to obtain the identity. We are only interested in the “owner tracing” aspect and not in the “coin tracing” aspect of fair e-cash. We shall express this phase as,

$\langle I, Proof_T \rangle := Trace(\mathcal{X}, \mathcal{T}, \langle A, Cert_A \rangle, \langle Proof_A \rangle, \{X_T\}_T)$, which should be read as, “ $\mathcal{X}$ engages in the tracing protocol with $\mathcal{T}$ using the values $\langle A, Cert_A \rangle$ and $\langle Proof_A \rangle$, to obtain the identity I and an *optional*¹ proof $Proof_T$, for proof of correct decryption of the ciphertext. The trustee uses its private key X_T for this purpose.”

3 Abstracting the Sealed Bid

In this section, we propose a mechanism for sealing the bid, which is central to the notion of the sealed bid auction procedure. We shall define the sealing process as follows:

Definition 1 *The sealing process is represented by:*

$$\langle Proof_S \rangle := Seal(b, r, I)$$

where $\langle Proof_S \rangle$ contains the sealed bid values along with the transcripts for proof of knowledge of the bid value, b , a randomiser, r , and the identity (or public key) of the sealer (or bidder), I . Given $\langle Proof_S \rangle$ the following must be true:

Hiding: *It must be intractable to determine the values of b or r .*

Binding: *It must be intractable to determine distinct tuples (b, r) and (b', r') such that,*

$$(\langle Proof_S \rangle := Seal(b, r, I)) \text{ AND } (\langle Proof_S \rangle := Seal(b', r', I))$$

Non-repudiation: *It must be intractable to determine (b, r, I) and (b', r', I') such that,*

$$(\langle Proof_S \rangle := Seal(b, r, I)) \text{ AND } (\langle Proof_S \rangle := Seal(b', r', I'))$$

unless $(b, r, I) = (b', r', I')$.

Clearly, the requirements for non-repudiation is a superset of the requirements for binding.

There may be many approaches to realise the sealed bid. The most prominent of them would be:

1. the signed commitment approach. Here the bidder can use a suitable commitment scheme [9,1] to commit to the bid and then sign the commitment value.
2. the signed encryption approach. Since semantically secure encryption schemes [4] can be idealised to be a commitment scheme, an encryption scheme can be used instead.

¹ It is surprising that many proposals have not explicitly concentrated on this aspect, namely proof of correct revocation.

The first approach can be used when unconditional confidentiality is a requirement for the sealed bid. If *revocation of confidentiality* from the sealed bid (that is without the participation of the bidder) the latter approach along with suitable *key escrow* techniques can be employed. In that case, the bidder can be expected to encrypt the value of the bid under the public key of a trusted entity. We shall employ the first approach in this paper.

3.1 A Concrete Proposal for the Sealed Bid

Based on the definition of the sealed bid we design a three pass, Schnorr type [12, 1] protocol to accomplish the desired goals.

System Settings A prime order subgroup G of $\mathbb{Z}_p^*$ is chosen to be of order q such that, $p = 2q + 1$ for sufficiently large prime p , so as to render the discrete logarithm problem intractable. Two generators, g and g_1 , for the group are published such that nobody knows² $\log_g g_1$. All operations are carried out in either $\mathbb{Z}_p^*$ or $\mathbb{Z}_q^*$ depending on the group being operated upon. The public key of the sealer (bidder) is *certified* to be $y_1 = g^{x_1}$ and $y_2 = g_1^{x_2}$.

Sealing Protocol An interactive protocol between the sealer and the receiver (of the seal) is as shown in Table 1. The sealer wishes to commit to the bid value

Table 1. The Sealing Protocol

Sealer	Receiver
$a, d_1, d_2 \in_R \mathbb{Z}_q^*$	
$S = g^b g_1^a, B = g^{d_1} g_1^{d_2}$	$\xrightarrow{S, B}$
	$\xleftarrow{c}$
$s_1 = d_1 - cx_1, s_2 = d_2 - cx_2$	$c \in_R \mathbb{Z}_q$
$t_1 = s_1 - bc, t_2 = s_2 - ac$	$\xrightarrow{t_1, t_2}$
	$B \stackrel{?}{=} (Sy_1 y_2)^c g^{t_1} g_1^{t_2}$

$b \in \mathbb{Z}_q$ and identify himself/herself using the public keys y_1 and y_2 . The sealer forms the commitment S to the bid value b and another commitment B for purpose of identification, and sends the two commitment values to the receiver. The receiver picks a random challenge c and returns challenge. The sealer then forms the response (s_1, s_2) with respect to the public key (y_1, y_2) and the commitment B . The sealer now uses (s_1, s_2) to respond to the commitment S as t_1 and t_2 . The idea behind this concept is that, the tuple (S, B, c, s_1, s_2) is unique (with

² When g_1 is the public key of a trusted entity or a Diffie-Hellman value of the public key of the trustee and the bidder, a similar approach can be used to design the signed encryption approach mentioned in the earlier section.

an overwhelming probability) in every protocol run and could not have occurred previously if the sealer or the verifier is honest. Therefore the responses t_1 and t_2 are unique in every protocol run. And so, the tuple (S, B, c, t_1, t_2) is also unique in every protocol run, with an overwhelming probability.

We present the following theorems for our proposal.

Theorem 1 *The proposed protocol belongs to the class of honest verifier zero-knowledge protocols.*

Proof: Clearly the protocol belongs to the three pass, honest verifier class. The protocol transcripts can be easily simulated by calculating $B = (Sy_1y_2)^c g_1^{t_1} g_1^{t_2}$ after choosing S, c, t_1 and t_2 . $\square$

Theorem 2 *If the values in the tuple (S, B, c, t_1, t_2) cannot be altered, then the protocol possesses the properties required for binding to the value of b .*

This theorem follows trivially from the theorem presented by Pedersen [9] (Theorem 3.1).

Corollary 1 *Given the tuple (S, B, c, t_1, t_2) , it will be infeasible to determine the value of b . Thereby, the protocol hides the value of b .*

Follows from Theorem 1.

Theorem 3 *When the sealer does not know the private keys corresponding to the public keys y_1 and y_2 , and the discrete logarithm problem is hard, the sealer convinces the receiver with a probability of $1/2^{|q|}$, where $|q|$ is the size of q in bits.*

Sketch: The sealer can cheat the receiver by guessing the challenge correctly in advance. Then by Theorem 1 the sealer can form correct transcripts. If $|q| = \log_2 q$, then the number of legal challenges will be of the form $2^{|q|}$. When the receiver chooses the challenges at random, as prescribed by the protocol, the probability that the sealer will correctly guess the challenge is $1/2^{|q|}$. $\square$

3.2 The Non-interactive Version

The interactive protocol suggested in Table 1 can be converted into a non-interactive version using the Fiat-Shamir heuristic [2]. For this purpose, we shall make use of a collision intractable hash function $\mathcal{H} : \{0, 1\}^* \mapsto \mathbb{Z}_q$. The sealer performs the following process with the bid b , his/her private key (x_1, x_2) and the commitment value b as the inputs to obtain the output as (S, t_1, t_2, c) .

Begin Process Sealer

Input: $\{x_1, x_2, b, a, g, g_1, p, q\}$
 $d_1, d_2 \in_R \mathbb{Z}_q^*$

Compute:

$$S = g^b g_1^a \bmod p, B = g^{d_1} g_1^{d_2} \bmod p$$

$$c := \mathcal{H}(y_1, y_2, S, B)$$

$$s_1 = d_1 - cx_1 \bmod q, s_2 = d_2 - cx_2 \bmod q$$

$$t_1 = s - bc \bmod q, t_2 = s - ac \bmod q$$

$$\text{Output: } \{S, t_1, t_2, c\}$$

End Process

The outputs of the sealing process can be verified by employing the following process:

Begin Process VerifySeal

$$\text{Inputs: } \{S, t_1, t_2, c, y_1, y_2, g, g_1, p\}$$

$$\text{If } c \stackrel{?}{=} \mathcal{H}(y_1, y_2, S, (Sy_1y_2)^c g^{t_1} g_1^{t_2} \bmod p), \text{ then}$$

$$\text{Result} \leftarrow \text{Pass}$$

Else

$$\text{Result} \leftarrow \text{Fail}$$

EndIf

$$\text{Output: Result}$$

End Process

In this process the verifier checks the sealing transcripts against the public key of the sealer.

To open the seal the sealer can release the tuples (b, a) . The values can be checked against the seal as follows:

Begin Process VerifyOpenedSeal

$$\text{Inputs: } \{a, b, S, t_1, t_2, c, y_1, y_2, g, g_1, p, q\}$$

$$\text{If } S \stackrel{?}{=} g^b g_1^a, \text{ then}$$

$$s_1 = t_1 + ac \bmod q, s_2 = t_2 + bc \bmod q$$

Else

$$\text{Result} \leftarrow \text{Fail}$$

$$\text{GoTo Output}$$

EndIf

$$\text{If } c \stackrel{?}{=} \mathcal{H}(y_1, y_2, S, (y_1y_2)^c g^{s_1} g_1^{s_2} \bmod p), \text{ then}$$

$$\text{Result} \leftarrow \text{Pass}$$

Else

$$\text{Result} \leftarrow \text{Fail}$$

EndIf

$$\text{Output: Result}$$

End Process

In this process the verifier checks the tuples (b, a) against the commitment value S . If they are correctly verified the actual signature value (s_1, s_2) is computed from t_1 and t_2 . The value of (s_1, s_2) is then checked for proper signature. Note that this is optional, because if the seal tuples pass the VerifySeal process and the tuple (b, a) are correctly verified against S , then (s_1, s_2) will be a legal signature tuple on S .

4 The Auction System

We shall now present a three phased auction system design using the electronic cash technology (see Section 2) and the process for sealing the bid proposed in Section 3.2.

System Settings: The system consists of a set of bidders $\mathcal{B}$, a mint, $\mathcal{M}$, for issuing electronic coins, a registrar, $\mathcal{R}$, an Auctioneer, $\mathcal{A}$, and a trustee $\mathcal{T}$.

A suitable prime-order subgroup, G of $\mathbb{Z}_p^*$, of order q , is chosen such that $p = 2q + 1$ is a large prime and the discrete logarithm problem is intractable. Suitable generators, g and g_1 , are chosen such that $\log_g g_1$ is not known to any entity. The arithmetic operations are performed in the relevant groups. A suitable hash function $\mathcal{H} : \{0, 1\}^* \mapsto \mathbb{Z}_q^*$ is chosen. Additional system setting requirements specific to the electronic cash technology are published along with tuple $(p, q, g, g_1, \mathcal{H})$.

The public key of the following entities are published:

1. The public key of each bidder, $I = g^{u_1}$, where u_1 is the corresponding private key.
2. The public key of $\mathcal{R}$, $y_r = g^{x_r}$, where x_r is the corresponding private key.
3. The public key of $\mathcal{A}$, $y_a = g^{x_a}$, where x_a is the corresponding private key.
4. The public keys of the mint, $\mathcal{M}$, $y_M = g^{x_M}$ and the trustee $f_T = g^{x_T}$.

4.1 The Three Phases

The pictorial representation of our model for auction system is presented in Figure 1. We propose a three phased schema for the design of an auction system. The three phases are:

1. Coin withdrawal phase, consists of the withdrawal sub-protocol;
2. registration phase, consists of the spend, **Sealer**, the **VerifySeal** sub-protocols; and,
3. bid submission phase, consists of the **VerifyOpenedSeal** sub-protocol.

We shall now discuss in detail each of these protocol phases.

Coin Withdrawal Phase: Bidder, $\mathcal{B}_i$ owning the public key I wishing to participate in individual auction activities, engages in the withdrawal protocol with the mint, $\mathcal{M}$, to obtain $\langle A_i, \text{Cert}_{A_i} \rangle$:

$$\langle A_i, \text{Cert}_{A_i} \rangle_I := \text{Withdraw}(I, M, \{s\}_I, \{X_M\}_M)$$

We refer to Section 2 for the interpretation.

Registration Phase: The bidder, $\mathcal{B}_i$, performs the following tasks:

1. presents the tuple (A_i, Cert_{A_i}) to $\mathcal{R}$;

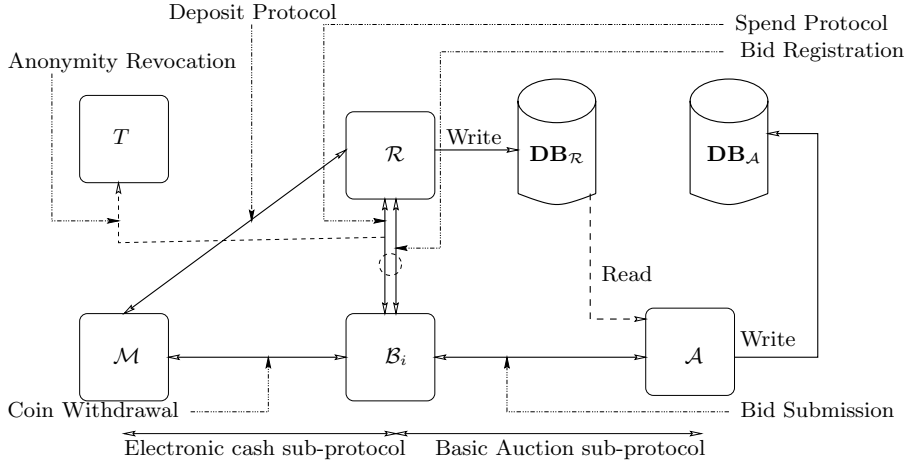


Fig. 1. System Dynamics of the Auction System

- engages in the spending protocol to convince its ownership of the tuple without revealing its identity. The spending protocol will be of the form:

$$\langle Proof_{A_i} \rangle := Spend(A_i, Cert_{A_i}, R, y_b, f_T, \{s, u_1\}_{A_i})$$

If the spending phase was successful, then $\langle Proof_{A_i} \rangle$ will contain $A_{1_i} = g^{u_1 s}$ and $A_{2_i} = g_1^s$ (see Section 2), such that $A_i = A_{1_i} A_{2_i}$.

- chooses its bid value, $b \in \mathbb{Z}_q^*$;
- seals the bid using the sealing process explained in Section 3.2. Towards this end it chooses $a \in_R \mathbb{Z}_q^*$ and computes the following:

$$\{S, s_1, s_2, c\} := Sealer(u_1 s, s, b, a, g, g_1, p, q)$$

Here $\{S, s_1, s_2, c\}$ are the outputs of the sealing process and $\{u_1 s, b, a, p, q\}$ are the inputs (see Section 3.2). If $\mathcal{R}$ verifies the sealing process correctly as:

$$Pass \stackrel{?}{=} VerifySeal(S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, g_1, p)$$

where $\{S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, h, p\}$ are the inputs to the process and the output is either pass for successful verification or fail for unsuccessful verification.

If $\mathcal{R}$ is satisfied with all the proofs in this section it signs the tuple as:

$$\sigma_{R_i} := Sign(x_r, S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, g_1, \langle Proof_{A_i} \rangle)$$

Where $Sign$ is a suitable signature algorithm that signs all the inputs

$$(S, s_1, s_2, c, g, g_1, A_{1_i}, A_{2_i}, \langle Proof_{A_i} \rangle)$$

using the private key x_r . $\mathcal{R}$ then stores:

$$(S, s_1, s_2, c, g, g_1, A_{1_i}, A_{2_i}, \langle \text{Proof}_{A_i} \rangle)$$

along with $\langle \text{Proof}_L \rangle$ and σ_{R_i} in a publicly readable directory $DB_{\mathcal{R}}$, indexed by A_{1_i} .

$\mathcal{R}$ *refrains* from accepting registration *after* a specified time, end of bid registration time (EBRT).

Bid Submission Phase This phase starts *after* the EBRT and finishes when the end of bid submission time (EBST) is reached, that is $\mathcal{A}$ accepts bids during the period between EBRT and EBST³. In this phase, $\mathcal{B}_i$ contacts the auctioneer, $\mathcal{A}$ and authenticates using its pseudonym A_{1_i} . $\mathcal{B}_i$ opens the commitment by sending (b, a) to $\mathcal{A}$. On receiving the tuple, $\mathcal{A}$:

1. obtains the registration transcripts from $DB_{\mathcal{R}}$ using A_{1_i} as the index into the database;
2. verifies the signature on the transcript by $\mathcal{R}$;
3. obtains the seal values, (S, s_1, s_2, c) , from the transcript;
4. verifies the opened commitments as;

$$Pass \stackrel{?}{=} \text{VerifyOpenedSeal}(a, b, S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, g_1, p, q)$$

and aborts the submission process when the result is not *Pass*;

5. signs the bid tuple (b, a) along with the the seal values (S, s_1, s_2, c) as:

$$\sigma_{\mathcal{A}_i} := \text{Sign}(x_a, S, s_1, s_2, c, A_{1_i}, A_{2_i})$$

where *Sign* is a suitable signature function, x_a the private key and $(S, s_1, s_2, c, A_{1_i}, A_{2_i})$ that is being signed to result in the signature tuple $\sigma_{\mathcal{A}_i}$;

6. returns the signature tuple $\sigma_{\mathcal{A}_i}$ to $\mathcal{B}_i$ as a receipt of the bids;
7. stores the tuples (b, a) along with $\sigma_{\mathcal{A}_i}$ and (A_{1_i}, A_{2_i}) in a publicly readable directory $DB_{\mathcal{A}}$, indexed by b .

Some bidders may refrain from opening their bids during this phase. If this is the case, the identity of such bidders can be traced using the anonymous token submitted, by the bidder, during the registration phase. Suitable action can then be taken against such bidders. Alternatively, a different method for the sealing process can be employed such that a quorum of trusted entities can recover the plaintext bid values *without the participation of the bidder*.

Announcement of Results: When the auction is announced to be terminated, the highest bid, b , is chosen from the database (which is publicly readable, thereby providing public verifiability) and $\mathcal{B}_i$ (the owner of the bid b) is announced as the winner. $\mathcal{B}_i$ identifies with $\mathcal{A}$ using the pseudonym, A_{1_i} , which is available in $DB_{\mathcal{A}}$ and avails the auctioned goods. Note that the anonymity of the winning bidder *need not* be revoked, but can be if necessary.

³ $\mathcal{R}$ is trusted not to register sealed bids after EBRT and $\mathcal{A}$ not to accept opened bids after EBST. This assumption is valid because $DB_{\mathcal{R}}$ and $DB_{\mathcal{A}}$ are publicly readable and can be suitably monitored for potential breach of trust.

Anomalies: There can be two cases of anomalies that could occur:

1. the winning bidder does not claim the goods; or,
2. the auctioned goods are denied to the winning bidder.

In the first case the winning bidder does not claim the goods and thereby does not pay for the goods. In which case, $\mathcal{A}$ or any other entity can approach the trustee $\mathcal{T}$ and engage in the tracing protocol to compute the identity, I , of $\mathcal{B}_i$, possessing the pseudonym, A_{1_i} . This is computed using the tracing protocol described in Section 2 as:

$$\langle I, Proof_T \rangle := Trace(X, T, \langle A_i, Cert_{A_i} \rangle, \langle Proof_{A_i} \rangle, \{X_T\}_T)$$

Note that all the information required for tracing are present in $DB_{\mathcal{A}}$ and $DB_{\mathcal{R}}$.

In the second case when the auction goods are denied to the winning bidder (due to software glitch or some other error), $\mathcal{B}_i$ can approach $\mathcal{R}$ with the receipt, $\sigma_{\mathcal{A}_i}$, that it received during the bid submission phase, identify itself using the pseudonym A_{1_i} and avail the goods or other compensations.

4.2 Analysis

In this section we shall verify the accomplishments of the protocol against the requirements stated in Section 1.1.

Confidentiality of bid: The confidentiality of the bid is provided by the hiding property of the sealing process, until the bid submission phase. Since, with an overwhelming probability, only the bidder can open the commitment values correctly, the scheme provides user-controlled confidentiality for the bid.

Non-repudiation of bid: This property is provided by the *non-transferability* property of the electronic cash scheme and the non-repudiation property of the sealing process. We stress the non-transferability property of the e-cash system because if the bidder transfers the power to spend the coin to another entity it would have to reveal the values of u_1s and s to that entity, and u_1 is a long term secret key of the bidder's account with the mint.

Publicly verifiable auction: Since all the proof transcripts in the system are publicly verifiable, the proposed auction system possess this property.

Anonymity of bidder: Conditional anonymity is provided to all the bidders using the e-cash system as an anonymous token issuer. Note that the anonymity of the winning bidder is also preserved.

Independence of auction rules: All the bid values, b , will reside in $DB_{\mathcal{A}}$ in *clear-text*. Any suitable auction rules can be employed to determine the winning bidder.

4.3 Comparison Based on Efficiency

In this section we shall compare the computational requirements of our scheme with the proposals using publicly verifiable secret sharing (PVSS) [13] schemes,

such as that of Franklin and Reiter [5], and the auction scheme proposed by Sakurai and Miyazaki [11]. The number of modular exponentiations by each entity for achieving confidentiality of bid were counted. The results are presented in Table 2.

An estimate (based on the publicly verifiable secret sharing scheme proposed by Schoenmakers [13]) of the number of modular exponentiations by each entity for achieving confidentiality of bid in schemes employing PVSS, such as that of Franklin and Reiter [5], is presented. We shall assume a t out of n scheme with $t = 2$ and $n = 2$, which is the simplest mode. The estimates are presented in Table 2.

The protocol proposed by Sakurai and Miyazaki [11] accomplishes anonymity of losing bidders and user controlled anonymity using undeniable signatures. The estimates assumes the following variables: L is number of bids, $J \in \{0, \dots, L-1\}$ is the index of winning bid value, J is the index of the winning bid and B the number of winning bidders. The assumed values are: $L = 10$ and $N = 100$. The estimates are presented in Table 2. In the table, the best case condition occurs when $J = 0$ and the worst case condition occurs when $J = 9$.

Table 2. Computational Comparison of Proposals

	Bidder	Auctioneer	Trustee
Our Scheme ^a	20	$5N/2N$	$15N$
PVSS schemes ^b	$4n + 2t$	$4nN$	$4(n + t)N$
Sakurai <i>et al.</i> ^c	$10J + 3$ (Losing Bidder) $10J + 13$ (Winning Bidder)	$6JN + 6B$	

^a Anonymity for winning and losing bidders
^b No anonymity for winning and losing bidders
^c Anonymity for losing bidders

The following observations are made on Table 2:

1. the bidders need to perform a constant number of exponentiations in our scheme;
2. the number of exponentiations that the auctioneer needs to perform:
 - a) is linear with the number of bidders, in our scheme;
 - b) is directly proportional to the number of bidders *and* the number of trustees in PVSS schemes; and,
 - c) is directly proportional to the *product* of the number of winning bidders and number of bids and, to the number of bidders, in [11].

Clearly, our scheme possesses a superior performance, in comparison with the approach based on the publicly verifiable secret sharing approach (without anonymity for losing bidders). In comparison with the scheme by Sakurai and Miyazaki, our scheme achieves the properties in a much more efficient manner with a constant number of exponentiations for the bidders.

4.4 Comparison Based on the Characteristics

We shall now present the characteristics of our scheme, in comparison with the other schemes, to demonstrate its achievements. The characteristics of our scheme are:

1. The number of exponentiations is a linear function of the number of bidders;
2. It is possible to provide anonymity even to the winning bidder;
3. Any type of auction rule can be employed, like highest price, lowest price or Vickery (second highest price).
4. Provides user controlled anonymity;
5. Any anonymity providing mechanism or sealing mechanism can be used, as long as they guarantee the required properties.
6. Phases 2 and 3 permit stateless operations. That is every bidder need not have continuous connections with the auction centre. This is very useful for implementations over stateless protocols like the HTTP protocol in the WWW applications on the Internet. Suitable anonymous token issuing facility can be employed to have a stateless Phase 1.

The characteristics of the schemes [5] that use publicly verifiable secret sharing are:

1. Users cannot control confidentiality of their bid during the bidding process;
2. Generally inefficient; and,
3. Independent of the auction rules.

The characteristics of the scheme proposed by Sakurai and Miyazaki [11] are:

1. It provides user controlled confidentiality for the bid values and the bid value of the losing bidders is not revealed.
2. It requires reliable real time networks and, therefore, may not be suitable for use over the Internet;
3. It can only operate with either the highest price or the lowest price auction rules, in order to provide anonymity for losing bidders;
4. Since, bidders must choose a value of the bid from a fixed set of bid values, it may not be suitable for all scenarios of auction;
5. The auction system requires critically on the state of the proceedings; and,
6. If the connection of *any single* bidder to the network is disconnected, due to some reason (accidentally or maliciously), the entire auction system will be stalled. Thereby, it is less robust.

Clearly our scheme is more practical, efficient and robust than the other schemes.

5 Conclusion

We presented a modular, publicly verifiable auction scheme that provides *user controlled anonymity*. The simple design, for a rather complex system, is due to

modular approach in the analysis and design methodology. The use of existing technologies, such as electronic cash, is crucial for designing complex systems with diverse properties.

The approach helped visualise the benefit of abstracting the sealing process, which can be further refined *without adversely affecting* the properties of the auction system. The sealing process can be improved to allow for the recovery of plain text bid values by a quorum of trusted parties, without the participation of the bidders. The concept of sealing is required in other application areas such as contract-signing, large scale voting, as well. This approach is ideally suited for voting protocols, as the identity of the participants can be guaranteed even after the third phase. Future research will be directed towards end.

References

1. Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In *Advances in Cryptology – CRYPTO’98*, pages 424–441. Springer, 1998.
2. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO’86*, pages 186–194. Springer-Verlag, 1986.
3. Yair Frankel, Yiannis Tsiounis, and Moti Yung. Indirect discourse proofs: Achieving efficient fair off-line e-cash. In *Asiacrypt’96*, 1996.
4. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
5. Matthew K. Franklin and Michael K. Reiter. The design and implementation of a secure auction service. In *IEEE Transaction on Software Engineering*, volume 22, pages 302–312, May 1996.
6. Michael Harkavy, Doug Tyger, and Hiroaki Kikuchi. Electronic auctions with private bids. In *Third USENIX Workshop on Electronic Commerce*, 1998.
7. Hiroaki Kikuchi, Michael Harkavy, and Doug Tygar. Multi-round anonymous auction protocols. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 1999.
8. Paul Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.
9. T. Pedersen. Non-interactive and information theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO’91*, 1991.
10. Kazue Sako. An auction scheme which hides the bids of losers. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, PKC’2000*, pages 422–432. Springer, 2000.
11. K. Sakurai and S. Miyazaki. A bulletin-board based digital auction scheme with bidding down strategy. In *Proceedings of CrypTEC’99*, pages 180–187. City University of Hong Kong Press, July 1999.
12. C.P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4:161–174, 1991.
13. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Advances in Cryptology – CRYPTO’99*. Springer-Verlag, 1999.
14. Markus Stadler. Publicly verifiable secret sharing. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT’96*, pages 190 – 199. Springer-Verlag, 1996.

An Online Public Auction Protocol Protecting Bidder Privacy

Khanh Quoc Nguyen¹ and Jacques Traoré²

¹ School of Computing and IT
University of Western Sydney, Nepean
`qnguyen@cit.nepean.uws.edu.au`

² France Télécom R & D
42 rue des Coutures, B.P. 6243
14066 Caen Cedex, France
`jacques.traore@francetelecom.fr`

Abstract. On-line auctioning is one of the fundamental services for the new Internet economy. Most current auction services are public auction where all bids are made available to any party. User privacy is a primary concern in electronic world because electronic environment facilitates the gathering of personal data. This paper proposes a public auction protocol that achieves bidder privacy using *binding* group signatures. A concrete solution for preventing defaults in auctions is also presented.

1 Introduction

On-line auction has appeared to be an integral part of the Internet economy. Many on-line auction servers such that **Yahoo!auction** and **eBay.com** have been successfully in operation. The number of on-line auctions has been growing exponentially. A check on 27 July 1999, saw over 2.4 million auctions running at **eBay.com** at the time.

On-line auction can be roughly classified into private auction and public auction. In private auction, each bidder secretly submits a bid before a deadline. No bidder should learn anything about the competing bids. At a certain time after the deadline, a winning bid is chosen by the auctioneer(s). Typical examples of private auction include auctioning of government contracts, art works and real estate. In public auction, all bids are made published and bidders are allowed to outbid competing bids. At the end of the auction, the highest bid wins. Public auction is often used in general auction where the secrecy of the bidders and bids are not a major concern.

In practice, most on-line auctions are public auction. This is the case in all auctions run at popular auction sites, e.g., **eBay.com**, **Yahoo!Auction** and **Amazon.com**. Current auction services provide weak level of security and privacy and assume a strong trustworthy assumption on the bidders. Any user can enrol in the system at will and can Bid any amount. If a winning bidder refuses to

pay, the only possible action is to conduct a new auction. Also the identity of any bidder is apparent to everyone.

Electronic environment facilitates the massive gathering of personal data and habits. Consequently, user privacy is a great concern in the on-line electronic marketplace. On-line auction is not an exception. In on-line auction, user privacy is represented in two forms: bidder privacy and bid secrecy. The difference is that bid secrecy means no party can know any bids other than the winning bid while bidder privacy allows the adversary to know all bids including losing bids but does not let the adversary find out the bid(s) of any bidder. In private auction, a bidder should not be able to learn any information about other competing bids, thus bid secrecy is critical. In public auction, the identities of bidders should be protected, thus bidder privacy is desirable.

This paper concerns on-line auction systems that protect bidder privacy. We present a novel public auction protocol that achieve bidder privacy using a new tool: a *binding* group signature scheme. The protocol provides privacy for all losing bidders and fairness to all parties involved. Particularly, the bidding history of any bidder that includes the winning bidder is not known to anyone. An adoption of this protocol for private auctions is also shown.

The remaining of this paper is organized as follows. Section 2 discusses the security requirements of on-line auction. Section 3 gives a brief summary of previous works existing in the literature. Section 4.1 presents the model of group signature and related tools that will be used as the underlying technique in our auction protocol. It includes a group signature sharing scheme. Section 5 describes our new auction scheme. The scheme is a public auction protocol that protects bidder privacy. Finally, Section 6 extends the results to private auctions.

2 Requirements

This section discusses common security requirements for on-line auctions. These requirements apply to public and private auctions. Main security issues of an on-line auction service are:

bid unforgeability: The bid(s) of a bidder can only be generated by the bidder herself. No party, even the ones who control the auction, can forge a bidder's bid.

result verifiability: The selection of the winning bid(s) is verifiable. The winning bid is dictated by the deterministic publicly known rule. No party can award the auction to another bid without being detected.

bid secrecy: Only the winning bid is made public, losing bids should remain secret.

bid unlinkability: It is infeasible to determine whether two bids are made by a same bidder.

bidder anonymity: The identity of bidders are protected. No one should learn about the identity of losing bidders.

Here we do not consider the issues that occur after the auction where the seller delivers the object(s) to the winning bidder for the correct payment. Such transactions can be done using standard fair exchange[2].

Bid secrecy, bid unlinkability and bidder anonymity come with two different flavors. One is to hold even after the auction is completed. The other is to hold until the selection procedure is taken place. The former is more desirable but the latter is acceptable in certain scenario.

Bid unforgeability and result verifiability are applicable to both public and private auctions. Bid secrecy is applicable only to private auction while bid unlinkability and bidder anonymity are applicable to privacy-protected auctions.

A private auction system should satisfy bid unforgeability, result verifiability, bid secrecy and bidder anonymity. Likewise a public auction system should satisfy bid unforgeability, result verifiability, bidder anonymity and bidder unlinkability.

Bid unlinkability is only a genuine concern in a private auction when multiple auctions are conducted at the same time by a same auction service. We say that an auction service achieves bidder privacy if it achieves both bid unlinkability and bidder anonymity.

3 Related Works

On-line auction has received considerable interests in security research community, mainly in the area of private auction, particularly in recent time.

Franklin and Reiter[12] proposed a private auction protocol using verifiable signature sharing and electronic cash. Their auction service is distributed over multiple non-related auctioneers. To submit a bid, the bidder generates a signature on the bid and distributes the signature and electronic coins to auctioneers using verifiable signature sharing. The bid is distributed to auctioneers using verifiable secret sharing. The secrecy of bids is achieved if no more than a certain number of auctioneers are faulty. In order to determine the winning bid, all bids are revealed during the selection process.

Harkavy, Tygar and Kikuchi[13] observed that this feature might not be desirable. They proposed a solution for this problem. In their protocol, bids are not actually revealed but compared to find the highest bid. They leveraged verifiable secret sharing and secret distributed computation in order to preserve bid secrecy. In their system, the bidders encode their bids into secrets that are distributed among auctioneers. After the deadline, the auctioneers compare the bids and eliminate bidders without revealing their bids or their identity using distributed multi-party computation. The winning bidder is revealed at the end of the computation. However, the price the authors have to pay, is that the computation and communication costs of auctioneers are enormous and the system can only practically tolerate up to a very small number of auctioneers. Moreover, the bidding range is limited and must be fixed before the auction.

Later, a different protocol using an oblivious third party was attempted by Cachin[3] on the same problem. The protocol is proposed for two-party case but

can be extended to a multi-party case. In this proposal, all bidders are required to involve in the selection process. The workload and communication complexity of each bidder grows linearly with the number of bidders. Also the identity of any bidder is known by all other (competing) bidders.

Under our security classification, the scheme in [12] achieves bid unforgeability, result verifiability and bid secrecy prior to the selection procedure. The schemes in [3,13] achieve bid secrecy even after the selection procedure. All of them do not provide bidder privacy. Note that, [12,13] provide limited bidder privacy, i.e., while the identity of a bidder is not known to competing bidders, each auction server knows the identity of all bidders at some stage during the auction.

All the above works have only considered the case of private auctions. They do not cover the case of public auctions. This is perhaps due to the perception that private auctions pose more challenging technical issues than public auctions. In our knowledge, there exists only few studies on public auctions in the literature[14,16]. However, those studies do not concern with the security aspect of public auctions in details but rather describe different methods of conducting public auctions.

Adopting existing private auction schemes to public auctions seems impossible. This is so because the main feature of existing works in private auction is to provide bid secrecy that is not applicable in public auctions.

4 Group Signatures

This section reviews the concept of group signatures and its extensions. These concepts are the building block for our auction protocol. Here, we present a verifiable group signature sharing scheme which is of independent interest and can be useful in many cryptographic protocols. In our knowledge, no such protocol exists in the literature.

A group signature scheme allows a group member to sign a message on the group's behalf such that everybody can verify the signature but no one can find out which group member has generated the signature. However, there is a trusted third party, called the revocation manager who can efficiently reveal the identity of the originator of a signature in the case of later disputes.

Formally, the model of group signatures consists of a group manager M , a revocation manager R , a set of group members $\mathcal{P} = \{P_1, \dots, P_m\}$, a security parameter k and five following procedures.

setup: A probabilistic protocol that on input of $O(1^k)$, outputs the group public key $\mathcal{Y}$, a *revocation secret key* ω to the revocation manager R and a *membership generation secret key* x to the group manager M .

registration: A probabilistic interactive protocol between a member P_i and the group manager M that on input of $O(1^k)$, outputs a *membership secret key* x_i to the group member P_i .

sign: A probabilistic protocol that on input of a message m , a membership secret key x_i and the group's public key $\mathcal{Y}$, $\text{sign}(m, x_i, \mathcal{Y})$ returns a signature σ of m .

verify: A deterministic algorithm that on input a message m , a signature σ and the group's public key $\mathcal{Y}$, $\text{verify}(m, \sigma, \mathcal{Y})$ outputs **yes** if and only if σ is a correct signature of m .

open: An algorithm that takes as input a message m , the group public key $\mathcal{Y}$, a signature σ and the revocation secret key ω . If σ is a valid group signature of m , $\text{open}(\sigma, m, \mathcal{Y}, \omega)$ outputs the identity of a member P_i and a proof that the group member P_i indeed signed m .

A group signature scheme is considered secure if it satisfies the following security requirements:

membership unforgeability: It is infeasible to compute a membership secret key x_i that can generate valid group signatures, without the knowledge of the membership generation secret key x .

signature unforgeability: Without a valid membership secret key x_i , it is infeasible to generate a signature σ on a message m .

anonymity: Given a signature, it is infeasible to find out which member has signed the message without the knowledge of the revocation key ω .

unlinkability: Given two signatures, it is infeasible to determine whether two signatures are issued by the same group member.

framing: Without the knowledge of the membership secret key x_i , no one can sign on the behalf of a group member P_i , even in cooperation with the group manager and the revocation manager.

revocation: Given a signature, the revocation manager who possesses the revocation key ω can always determine the member who issued the signature.

Historically, the concept of group signatures were introduced by Chaum and van Heyst[9]. Since its invention, several concrete, improved and generalized group signature schemes were proposed[10,4]. However, all these schemes are inefficient and require the size of group public key and group signatures to grow linearly with the size of the group. Recently, Camenisch and Stadler proposed two efficient group signature schemes in [6] that achieve constant size for the group public key and group signatures. Later, group signature schemes with improved performance and better flexibility were proposed in [5,7,18]. We note here that the revocation manager in all those schemes can be shared among a set of entities using verifiable secret sharing[17]. This feature is critical to provide efficient robustness in our auction protocol.

4.1 A Group Signature Scheme

For concreteness, this subsection reviews one of the group signature schemes presented in [6]¹. Since the description of the scheme is rather complex, we will focus on the features that we will need for our construction of the auction system.

¹ In fact, we will use a modified version of this scheme in order to thwart the attack described in [18].

The System Setup Protocol. To setup the group signature scheme, the group manager M computes the following information:

- an RSA public key (n, e) , where n is the product of two safe primes,
- a cyclic group $G = \langle g \rangle$ of order n over the finite field $\mathbb{Z}_p$ for a prime p ,
- an element $a \in \mathbb{Z}_n^*$ that is of the order $\phi(n)/4$ and
- an upper bound λ on the length of the secret keys and a constant $\epsilon > 1$.

The revocation manager R computes the followings

- a secret key $\omega \in_R \mathbb{Z}_n^*$,
- a random generator h of G ,
- $y_R = h^\omega$ and
- a constant $b \neq 1$.

The group public key then is $\mathcal{Y} = (n, e, G, g, a, \lambda, \epsilon, h, Y_R)$. The group manager private key (i.e, the membership generation secret key) is d , which is the corresponding secret key of the RSA public key (n, e) and the revocation manager's private key is ω .

The Registration Protocol. To register as a group member, P_i chooses a random secret number $x \in_R \{0, \dots, 2^\lambda - 1\}$ and sends the value $y = a^x \bmod n$ to the group manager. In turn, the manager returns $v = (y + b)^d \bmod n$. The membership secret key then is (x, v) satisfying:

$$a^x + b = v^e \bmod n.$$

Here x is the secret part of the membership secret key that is not seen by the group manager.

The Sign & Verify Protocols. Given a message m , a group member who possesses a membership secret key $(x, v, y = a^x)$ can generate a group signature σ on m as follows:

- $\tilde{g} = g^r$ and $\tilde{z} = \tilde{g}^y$ for $r \in_R \mathbb{Z}_n^*$,
- $d_1 = y_R^u g^y$ and $d_2 = h^u$ for $u \in_R \mathbb{Z}_n^*$,
- $V_1 = \text{SK}\{(\gamma, \delta) : \tilde{z} = \tilde{g}^\gamma \wedge d_2 = h^\delta \wedge d_1 = y_R^\delta g^\gamma\}(m)$,
- $V_2 = \text{SK}\{(\beta) : \tilde{z} = \tilde{g}^{a^\beta}\}(V_1)$, and
- $V_3 = \text{SK}\{(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}\}(V_2)$.

The verification is to check the correctness and consistency of the signatures of knowledge V_1 , V_2 , and V_3 .

Note: Here, $\text{SK}\{(x_1, \dots, x_k) : z_1 = f_1(x_1, \dots, x_k) \wedge \dots \wedge z_l = f_l(x_1, \dots, x_k)\}(m)$ denotes a signature of knowledge on the message m of the secrets $x_1, \dots, x_k$ satisfying all l statements: $z_1 = f_1(x_1, \dots, x_k)$, ..., and $z_l = f_l(x_1, \dots, x_k)$. The main character of signatures of knowledge is that a signature of knowledge cannot be constructed without the knowledge of the secret(s). A detailed discussion about signatures of knowledge can be found in [4].

The Open Protocol. To open a signature, the revocation manager computes $z = d_1/d_2^\omega = g^y$. Then z is matched with all the values of g^y stored in the database of registered group members. As y is seen by the group manager in the registration, g^y can be computed and stored. The signer is identified once a match is found.

So far, this group signature scheme is considered secure. Though, no formal proof is known yet.

4.2 A Verifiable Group Signature Sharing Scheme

Apart from a group signature scheme we further need a new tool, namely a verifiable group signature sharing scheme.

The concept of signature sharing was introduced in [11]. In this technique, the showing of a signature is first converted to the showing of some public information and the showing of a particular homomorphic inverse. Intuitively this means that to reveal a signature one can simply show some public information and the inverse of some value under a homomorphic function. Then the homomorphic inverse is shared amongst the designated entities using secret sharing. We now show a group signature sharing scheme based on the group signature scheme given in the previous section.

Before going further, we give the detailed construction of the signature of knowledge $SK[(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}]$ that is used in the group signature scheme. A $SK[(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}](V_2)$ consists of the tuple (c, s) satisfying the equation

$$c = \mathcal{H}(V_2 \| g_0 \| \dots \| g_e \| g_0^r \| \dots \| g_{e-1}^r),$$

where r is a random number, $s = r - c\alpha \bmod n$, $g_0 = \tilde{g}$ and $g_{i+1} = g_i^\alpha$ for $(i = 0, \dots, e-1)$. The signature of knowledge can be verified by checking whether $\tilde{z}\tilde{g}^b \equiv g_e$ and

$$c = \mathcal{H}(V_2 \| g_0 \| \dots \| g_e \| g_0^s g_1^c \| \dots \| g_{e-1}^s g_e^c),$$

Let $y_i = g_i^s$ for $i = 0, \dots, e-1$. This signature of knowledge can be reduced to the showing of the public information $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ and a secret information s . Here $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ satisfies

$$c = \mathcal{H}(V_2 \| g_0 \| \dots \| g_e \| y_0 g_1^c \| \dots \| y_{e-1} g_e^c).$$

Lemma 1. *A set of $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ that satisfies*

$$c = \mathcal{H}(V_2 \| g_0 \| \dots \| g_e \| y_0 g_1^c \| \dots \| y_{e-1} g_e^c),$$

can be constructed from public information.

Such a set $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ can be created from public information by choosing $w_0, \dots, w_{e-1}$ at random, computing,

$$c = \mathcal{H}(V_2 \| g_0 \| \dots \| g_e \| w_0 \| \dots \| w_{e-1})$$

and setting $y_i = w_i/g_{i+1}^c$ for $i = 0, \dots, e-1$. To determine whether $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ presents a valid signature, one has to determine whether $\log_{g_i}(y_i) \equiv \log_{g_j}(y_j)$ which is hard for $i \neq j \in \{0, \dots, e-1\}$.

Theorem 1. *In this construction, the showing of a signature*

$$\text{SK}\{(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}\},$$

is equivalent to the showing of some public information $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ satisfying

$$c = \mathcal{H}(V_2 \| g_0 \| \dots \| g_e \| y_0 g_1^c \| \dots \| y_{e-1} g_e^c),$$

and a secret s such that $y_i = g_i^s$ for $i = 0, \dots, e-1$.

Proof (Sketch). It is trivial to see that a signature $\text{SK}\{(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}\}(V_2)$ can be converted into $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ and s . Also given a public information $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$, if there is a value s such that $y_i = g_i^s$ for $i = 0, \dots, e-1$ then $(c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ and s form a valid signature.

Next, we show how to convert the showing of a group signature to the showing of some public information and the showing of a discrete logarithm. This is a critical step in our group signature sharing scheme. A group signature σ consists of

- $\tilde{g} = g^r$ and $\tilde{z} = \tilde{g}^y$ for $r \in_R \mathbb{Z}_n^*$,
- $d_1 = y_R^u g^y$ and $d_2 = h^u$ for $u \in_R \mathbb{Z}_n^*$,
- $V_1 = \text{SK}\{(\gamma, \delta) : \tilde{z} = \tilde{g}^\gamma \wedge d_2 = h^\delta \wedge d_1 = y_R^\delta g^\gamma\}(m)$
- $V_2 = \text{SK}\{(\beta) : \tilde{z} = \tilde{g}^{a^\beta}\}(V_1)$, and
- $V_3 = \text{SK}\{(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}\}(V_2)$.

We keep d_1, d_2, V_1 and V_2 and divide V_3 to $\mathcal{P} = (c, V_2, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ and $\mathcal{S} = (s)$. Then to show a valid signature σ , one can show the public information $\{(d_1, d_2, V_1, V_2), \mathcal{P}\}$ and the discrete logarithm s . As $\mathcal{S}$ and $\mathcal{P}$ forms a valid V_3 , s and $\{(d_1, d_2, V_1, V_2), \mathcal{P}\}$ forms a valid group signature. It remains to show that $\{(d_1, d_2, V_1, V_2), \mathcal{P}\}$ can be computed from public information. This is done as follows. First choose random numbers $r \in_R \mathbb{Z}_n^*$ and $\beta \in \{0, \dots, 2^\lambda - 1\}$. Then compute $y = a^\beta$, $\tilde{g} = g^r$, $\tilde{z} = \tilde{g}^y$. Next choose $u \in_R \mathbb{Z}_n^*$ and compute $d_1 = y_R^u g^y$ and $d_2 = h^u$. Now form $V_1 = \text{SK}\{(\gamma, \delta) : \tilde{z} = \tilde{g}^\gamma \wedge d_2 = h^\delta \wedge d_1 = y_R^\delta g^\gamma\}(m)$ and $V_2 = \text{SK}\{(\beta) : \tilde{z} = \tilde{g}^{a^\beta}\}(V_1)$. Finally compute $\mathcal{P} = (c, g_0, \dots, g_e, y_0, \dots, y_{e-1})$ as in the converting algorithm for $\text{SK}\{(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}\}(V_2)$. It is clear that V_1 and V_2 are valid signatures of knowledge. Thus showing a valid signature σ , is equivalent to showing some public information and a discrete logarithm s satisfying $y_i = g_i^s$ for $i = 0, \dots, e-1$.

Now to share a group signature σ between several entities, the user who possesses the signature, divides the signature into some public information $\{(d_1, d_2,$

V_1, V_2), $\mathcal{P}$ and the discrete logarithm s . She then sends all the public information to all entities and distributes the shares of the discrete logarithm s to the respective designated entities using secret sharing. The group signature sharing is verifiable if verifiable secret sharing is used. As discrete-logarithm based (verifiable) secret sharing is a well-studied topic, we omit the constructions.

5 A Public Auction Protocol Protecting Bidder Privacy

This section presents a public auction system that protects bidder privacy. In a high level description, the auction system works as follows. We deploy a group signature scheme in that the group manager is the auction manager and auction participants (sellers and buyers) are group members. To enrol in the system, each participant contacts the group manager and obtains a membership secret key. Then for each auction, a bidder B_i can bid a value by generating a signature on the bid using her membership secret key x_i . At the deadline, the identity of the bidder, who posts the highest bid, is retrieved using the revocation procedure of the group signature scheme. Bidder privacy is protected due to the anonymity and unlinkability properties of the underlying group signature scheme.

Formally, the auction system works as follows. Let M be the auction manager, $\mathcal{A}$ the auction server, $S_1, \dots, S_m$ the sellers, and $B_1, \dots, B_l$ the bidders in the auction service. The auction manager is the party who organize the auction and its main task is to set up the auction service. The auction server is the party who conduct auctions. Sellers are those who wish to run some items through the auction service. Bidders are those who wish to buy some items through the auction service. In practice, M and $\mathcal{A}$ are likely played by a same party. However we use different notations to distinguish their roles.

Similar arguments apply for the notations of S_j and B_i . We use S to denote a generic seller, P to denote a generic participant, i.e., a seller or a bidder. Further we assume the existence of two bulletin boards $\mathcal{B}$ and $\mathcal{L}$. $\mathcal{B}$ is publicly readable/writable but only $\mathcal{A}$ can delete the content already posted in $\mathcal{B}$. $\mathcal{L}$ is publicly readable by everyone but only $\mathcal{A}$ can modify the content of $\mathcal{L}$.

For the moment, we assume that $\mathcal{A}$ and respectively M are implemented as a single *trusted entity*. Later we will discuss how to share the functionalities of $\mathcal{A}$ and M over a quorum of non-related auctioneers and managers.

The auction service consists of the five following procedures:

setup: To setup the service, M announces the service and its scope. Further, M identifies the identity of $\mathcal{A}$. M and $\mathcal{A}$ then run **setup** procedure of a group signature scheme which generates the service public key $\mathcal{Y}$, a service secret key x known to only M and a group revocation secret key ω known only to $\mathcal{A}$.

registration: To participate into the system, each party P has to register with M . If M accepts P into the service, M issues P with a certificate **cert** stating the conditions of P membership. This might include whether M allows P to be a seller/buyer and/or in which capacity. Also M runs **registration** procedure

of the group signature scheme with P by that P obtains a membership secret key x_P that is the membership secret key of the group signature scheme.

auction preparation: To auction an item, S contacts M using her/his membership certificate with the description of the item. If the description is within the conditions of S 's membership certificate, M signs the description, creates a thread on $\mathcal{B}$ for S and posts the item's description with its signature on the item on the thread. Also S informs $\mathcal{A}$ of the auction.

bid submission: If a bidder B_i wishes to bid for an item, B_i checks the item's thread, verifies that the auction is genuine, i.e., authorized by M . Then B_i posts his/her bid on the item on the thread by running the **sign** procedure of the group signature using his/her membership secret key.

bid selection: After the deadline, $\mathcal{A}$ chooses the highest (or the most suitable) bid and runs the **verify** procedure. If the output is **yes**, $\mathcal{A}$ accepts the bid as the winning bid. Otherwise, $\mathcal{A}$ repeats the process for the remaining bids. Once a winning bid is determined, $\mathcal{A}$ runs the **open** procedure with the concerning signature (i.e., the winning bid) and identifies the winning bidder B_j . In this phase, it is also essential for $\mathcal{A}$ to verify that the winning bidder remains valid by checking the black list $\mathcal{L}$. If the answer is **no**, $\mathcal{A}$ repeats the process with the remaining bids.

winner announcement: After the winning bid is chosen and verified, $\mathcal{A}$ posts the identity of the winning bidder on the thread along with the identities of bidders for all failed higher bids. $\mathcal{A}$ should include some proofs that the failed bids are made by those bidders. Then result verifiability is achieved by checking whether failed bidders are in the black list $\mathcal{L}$.

5.1 Security

We show that the proposed auction system achieves bid unforgeability, result verifiability, bid unlinkability and bidder anonymity. As each bid is a group signature, bid unforgeability, bid unlinkability and bidder anonymity come from the security properties of the underlying group signature scheme. Result verifiability is due to the deterministic rule of the selection process and that the identities of bidders, who submitted higher bids, are published. Thus one can verify the correctness of the auction by checking whether the winning bid is chosen according to the rule and whether all the identities of bidders, who make better bids, are in the black list $\mathcal{L}$.

5.2 Other Issues

The black list $\mathcal{L}$. $\mathcal{L}$ is the list of participants that have their memberships revoked. A membership is usually revoked when the participant is detected to commit a fraud. At start up, $\mathcal{L}$ is empty. M puts a participant P on the blacklist $\mathcal{L}$ if P has refused to pay or to deliver in some previous auctions. Frauds are not actively detected but found when other participants complain and the evidences

are justified. Here we assume that complains on the transactions between winning bidders and sellers can be verified. In later sections, we discuss this issue further.

Clearly, the size of $\mathcal{L}$ will grow with time. It is essential to limit the size of $\mathcal{L}$. It is done by expiring the public key $\mathcal{Y}$ after a certain period, thus requiring each participant to renew their secret membership key and updating public information. Alternative method is to incorporate partial blind signature[1] into group signature model. The advantage then is that public information can be left unchanged for a longer period.

Distributing the role of $\mathcal{A}$ and $\mathcal{M}$. The main role of $\mathcal{M}$ is to enrol sellers and bidders into the system. $\mathcal{M}$ is not actively involved in any auction. In fact, the security and privacy of any auction do not rely on $\mathcal{M}$. Furthermore $\mathcal{M}$ is the owner of the service, there is not much incentive for $\mathcal{M}$ to cheat. Thus using a single party to play the role of $\mathcal{M}$ is reasonable. Of course, $\mathcal{M}$ can be distributed among several entities. Then the registration procedure uses distributed multi-party computation which involves intensive computation. However, it is still acceptable since the procedure is one-off in regard to the sellers and bidders. In this case, $\mathcal{M}$ should only be distributed over a small number of entities. Note that schemes in [3,13] require some form of multi-party computation in every auction.

The main role of $\mathcal{A}$ is to identify the winning bidder in each auction, which is the role of the revocation manager in the group signature scheme. Subsequently, $\mathcal{A}$ can be distributed among several entities using verifiable secret sharing. The computation cost for each entity is scalable and reasonable. It is only linearly dependent on the number of entities, not on number of bids.

Here we emphasise that bidder privacy is still protected against any number of faulty servers or managers under the threshold. This is not achieved in the protocols in [12,13,3].

User Privacy. Users are all parties that participate in the system either as sellers or bidders. Only a small subset of users take part in each auction. To register in the system, each user runs the registration procedure to obtain a certificate from $\mathcal{M}$. In this procedure, it is logical that $\mathcal{M}$ has to know the user's identity so that $\mathcal{M}$ can determine whether to issue a certificate to the user. In return for the certificate, $\mathcal{M}$ should receive some evidence that legally binds the user to the system policy. Otherwise, it is not legally possible to hold a dishonest user accountable. In reality, the evidence is the user's agreement on the system policy. Such an agreement is often represented as a signature of the user on the system policy. In some circumstances, a dishonest $\mathcal{M}$ might compromise user privacy by passing the agreements to other entities, e.g., for advertising purposes. Thus, it is desirable to have a mechanism that legally bind the user to the system policy while preventing $\mathcal{M}$ to give these information to others. Here, we present the intuition of such a solution.

Before going further, let us repeat the construction of group signature schemes. When a member is enrolled in the group, the group manager (i.e., $\mathcal{M}$ in our

case) issues a membership secret key to the member. This membership secret key is a membership certificate that can be issued only by the group manager. In order to prevent framing attacks, the group manager does not see the whole certificate. The certificate contains some information that is only known to the member. This information is used in group signature generations.

Assuming that each user P has a legal binding secret/public key pair $(sk, pk = f(sk))$ where $f()$ is some hard-to-inverse function, we design the solution as follows. In the registration, P does not have to sign any agreement but P is required to use sk as a part of the certificate that is not seen by M in the membership secret key. Of course, P has to prove to M this fact in (non-transferable) zero-knowledge.

It is difficult for M to prove to others that P is a user because M would not be able to demonstrate that the secret key sk of P is used in the membership secret key. However when P misuses the membership secret key, M can prove that in the court once M detects the key misuse. The reason is that in this case, M detects the misuse only when M gathers a signature signed with the membership secret key of P . A signature can be signed only with the full knowledge of a membership secret key. Thus when P signs a message, P has to use sk . This value is not known to M . Then in the court, M can prove that P is the misbehaving user by asking P to prove whether sk is a part of the membership secret key used to generate the signature. If P can prove the contrary then M is the cheating party. Note that only P can carry out the proof because P knows sk .

It remains to show a protocol that allows an user to prove that her secret key is included in her membership secret key of the group. We proceed by giving such a protocol for the group signature scheme given in the Section 4.1. We assume that each user has a legally-bound pair of secret/public key which can be for example a key pair $(x, w = G^x \bmod P)$, where G is a generator of a subgroup of prime order Q over the prime finite field P . The values of G, Q and P are chosen by some certificate authority that is *completely* independent of the group signature system.

Now each user can prove to the group manager that the secret information generated x by the user and kept secret from the group manager, is the same as secret key X of the user's legally binded key pair $(X, W = G^x)$ using the following binding protocol.

Let l denote the size of the challenge chosen by the group manager. The value of l must not be too large. Typically l is chosen as $l = 40$. We further assume that the value λ satisfying $2^{\epsilon(\lambda+l)+2} < Q$. First the group manager sends to the user a large number N that is a product of two safe prime numbers and the user has no prior knowledge of its factoring. The number N should satisfy $|N| > (2|P| + 1)$ and $|N| > (2|n| + 1)$. If necessary, the group manager can prove the correctness of N using the protocols given in [15]. The group manager also sends a number A which is a quadratic residue modulo N that has a large order in $\mathbb{Z}_N^*$, to the user. The binding protocol is as given in Figure 1.

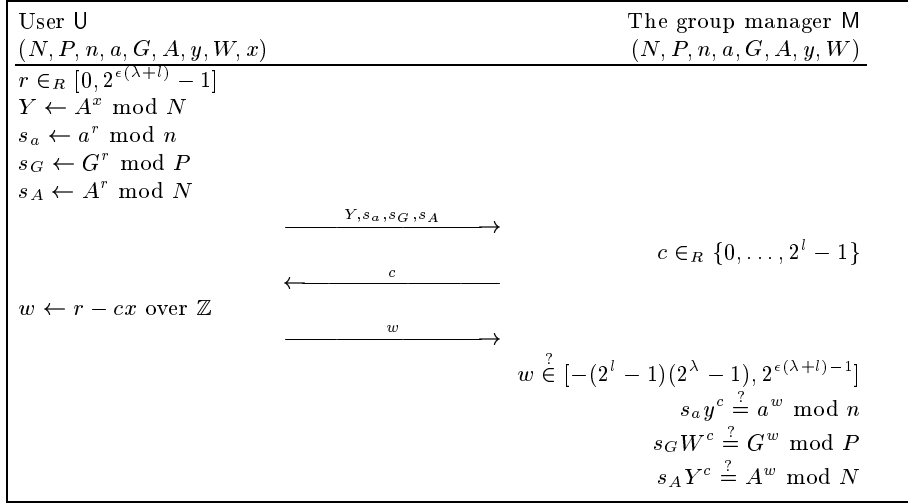


Fig. 1. The binding protocol

For a discussion of the security of this scheme, we refer the reader to [8,18] (where the security of a similar protocol is analyzed). We also refer to [18] for a method to prevent abuse made with compromised group signature keys.

If the group manager accuses the user of cheating, e.g., to default her winning bid, the user who possesses a legally-binded public key $\tilde{Y} = G^{\tilde{x}}$, can prove her honesty by showing $\tilde{Y}$, $\tilde{y} = a^{\tilde{x}}$ and that the same $\tilde{x}$ is used in $\tilde{Y}$ and $\tilde{y}$. Then the user is innocent if $\tilde{y} \neq y$.

5.3 Default Prevention

So far, we have only considered security issues occurred prior the winning bid is chosen. In this section, we consider the issues occurred after the winning bid is selected. The main security concern after the winning bid is awarded, is the case where the winning bidder defaults on her bid. So far we propose that in this case, the system manager can put the bidder on the black list $\mathcal{L}$ once the default is detected. Nonetheless, in such cases, the seller has to organize a new auction altogether. Ideally, there should be some way that can enforce the bidder to pay once the bidder is awarded the auction.

We now present a solution using on-line anonymous cash. It is a modification of the solution presented in [12]. The solution works for both private and public auctions. The idea is that when submitting a bid, each bidder also submits commitments of coins that value not less than the bided amount. As coins are anonymous, bid secrecy and bidder privacy are maintained.

The commitment scheme is constructed so that each commitment uniquely identifies a coin but the coin cannot be constructed from the commitment. Then when a winning bid is chosen, the seller contacts the bank who issued the coins,

to verify the availability of the coins. As each commitment uniquely identifies a coin, the bank can check the availability of the coins. If the coins are not spent, the seller awards auction to the bidder. In this case, the coins are reserved, no longer available. Default prevention is then achieved as the seller can always claim the coins, i.e., receives the payment by either from the user in normal case or from the bank. In the latter case, the seller sends the goods to the bank who forwards them to the user.

In order to prevent the seller to illegally reserve the coins without actually sending the goods, the user can clear the coins by asking the financial institution to act as the user's agent in the transaction with the seller. Then the financial institution requests the seller to send the goods. If the goods are not received within a reasonable time frame, the financial institution clears the coins and cancels the transaction. It remains to show the construction of the commitment of the coins.

In anonymous cash schemes, coins are represented as signatures. For the sake of simplicity, we assume that the signature scheme in use, is Schnorr signature scheme. Other signature schemes work basically in the same manner.

In the Schnorr signature scheme, a public key is generated by choosing two prime numbers of suitable length p and q such that $q|(p-1)$. A signature on the message m is then the tuple $\{c, r\}$ that can be verified by checking that $c = \mathcal{H}(g^r y^c || m)$.

The showing of the Schnorr signature $\{c, r\}$ on the message m is *equivalent* to showing $\{c, u = g^r\}$ and the promise of the homomorphic inverse $r = \log_g(u)$, where $\{c, u\}$ satisfies $c = h(uy^c || m)$. Note that without knowing the private key, one can compute a valid pair of (c, u) by choosing $z \in Z_q$ at random and calculating $c = \mathcal{H}(y^z || m)$ and $u = y^{z-c}$. However, it is not possible to know the inverse of u in these cases.

Then to commit a coin which is the signature (c, r) , the user sends (c, u) and generates a Schnorr signature σ on a message that uniquely identifies the auction, using the public/secret key $(u = g^r, r)$. The signature σ demonstrates the knowledge of r thus the commitment demonstrates knowledge of the coin. As c is included in both commitment of the coin and the coin itself, the commitment uniquely identifies the coin. However as computing r from σ is hard, it is not possible to compute the coin from its commitment.

6 Providing Bidder Privacy in Private Auction Schemes

Existing private auction schemes do not explicitly provide bidder privacy. In this section, we use group signatures to provide bidder privacy for the schemes in [12, 13]. Providing bidder privacy for the scheme in [3] is difficult if possible. This is so because bidders have to exchange information with the others in winning bid selection procedure; thus naturally they should know the identity of competing bidders.

Group signatures can provide bidder privacy for private auction schemes in [12,13]. The modification is that instead of signing with normal signature, each

bidder generates a group signature on her bid. Then, each bidder must register to the system beforehand as in our auction scheme. A group signature on a bid is distributed using verifiable group signature sharing described earlier. The bid selection remains the same, i.e., using distributed multi-party computation[13] or opening all bids[12]. Once the winning bid is determined, the auctioneers together open the bid and identify the bidder.

Due to unlinkability feature of group signatures, the modified private auction schemes also provide bidder privacy in multi-round private auctions where after each round, the highest bid is announced and bidders are allowed to bid higher bids in another round. The winning bid is chosen after the final round.

References

1. M. Abe and E. Fujisaki. How to date blind signatures. In *Advances in Cryptology - ASIACRYPT'96*, pages 244–251. LNCS vol. 1163, Springer-Verlag, 1996.
2. N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1998.
3. C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of 6th ACM Conference in Computer and Communications Security*. ACM Press, 1999.
4. J. Camenisch. Efficient and generalized group signatures. In *Advances in Cryptology - EUROCRYPT'97*, pages 410–424. LNCS vol. 1233, Springer-Verlag, 1997.
5. J. Camenisch and M. Michels. A group signature scheme based on an rsa-variant. Technical Report RS-98-27, BRICS, 1998. a preliminary version of this paper appeared in *Advances in Cryptology - ASIACRYPT'98*.
6. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology - CRYPTO'97*, pages 410–424. LNCS vol. 1296, Springer-Verlag, 1997.
7. Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology - CRYPTO'99*, pages 106–121. LNCS vol., Springer-Verlag, 1999.
8. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. In *Advances in Cryptology - Eurocrypt'98*, pages 561–575. LNCS vol. 1403, Springer-Verlag, 1998.
9. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT'91*, pages 256–265. LNCS vol. 547, Springer-Verlag, 1991.
10. L. Chen and T.P. Pedersen. New group signature schemes. In *Advances in Cryptology- EUROCRYPT'94*, pages 171–181. LNCS vol. 950, Springer-Verlag, 1995.
11. M. Franklin and M. Reiter. Verifiable signature sharing. In *Advances in Cryptology - Eurocrypt'95*, pages 50–63. LNCS vol. 921, Springer-Verlag, 1995.
12. M. Franklin and M. Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5):302–312, 1996.
13. M. Harkavy, D. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *Proceedings of 3rd USENIX Workshop on Electronic Commerce*. Boston, 1998.
14. M. Kumar and S. Feldman. Internet auctions. In *Proceedings of 3rd USENIX Workshop on Electronic Commerce*. Boston, 1998.
15. Tri Van Le, Khanh Quoc Nguyen, and Vijay Varadharajan. How to prove that a committed number is prime. In *Advances in Cryptology - Asiacrypt'99*, pages 208–218. LNCS vol. 1716, Springer-Verlag, 1999.

16. T. Mullen and M. Wellman. The auction manager: Market middleware for large-scale electronic commerce. In *Proceedings of 3rd USENIX Workshop on Electronic Commerce*. Boston, 1998.
17. T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO'91*, pages 129–140. LNCS vol. 576, Springer-Verlag, 1992.
18. Jacques Traoré. Group signatures and their relevance to privacy-protecting off-line electronic cash systems. In *Information Security and Privacy (ACISP'99)*, pages 228–243. LNCS vol. 1587, Springer-Verlag, 1999.

Algorithms to Speed Up Computations in Threshold RSA^{*}

Brian King

¹ University of Wisconsin-Milwaukee, Milwaukee, WI 53201 USA

² Motorola Labs, 1301 Algonquin Rd, Schaumburg IL 60196, USA
king@labs.mot.com, brking@execpc.com

Abstract. A t out of l threshold scheme is such that shares are distributed to l participants so that any set of t participants can compute the secret, whereas any set of less than t participants gain no information about the secret. In [9], Desmedt and Frankel described a threshold scheme which can be used with any finite abelian group. Hence it can be used to provide a threshold RSA scheme. In their scheme, a share size is $O(l) \cdot$ “size of secret”. Each shareholder performs $O(t^3 l^2 (\log_2 l)^2)$ elementary operations and $O(l t \log_2 l)$ group operations. Here, we describe an algorithm which will reduce the amount of elementary operations, so that each shareholder performs $O(t^2 l \log_2 l)$ elementary operations and $O(t \log_2 l)$ group operations.

Keywords: Threshold secret sharing, Threshold RSA, cyclotomic polynomials.

1 Introduction

A t out of l threshold scheme is such that shares, formed from a secret k and random elements, are distributed to l participants so that any set of t participants can compute k , and where any subset of $t - 1$ or less participants gain no information about k . If $s_1, \dots, s_l$ represent the shares distributed to the l participants

- $Prob(k | s_{i_1}, \dots, s_{i_t}) = 1$
- $Prob(k | s_{i_1}, \dots, s_{i_{t-1}}) = Prob(k)$.

RSA is an important cryptographic scheme. The development of threshold RSA was problematic due to the fact that the modulus $\phi(N)$,^{1 2} as well any multiple, cannot be leaked to any of the shareholders. Threshold RSA has been examined in [11,7,8], then in [14,9,6], and most recently in [16,13,12,19]. Of particular interest to us is the scheme by Desmedt, Frankel et. al. [9,6]. Here the authors developed a threshold scheme that can be used with any finite abelian

^{*} This work was partially funded by NSF grant CCR-9508528

¹ Here $N = pq$, the product of two distinct primes, and $\phi(N) = (p - 1) \cdot (q - 1)$.

² The true modulus is the Carmichael function $\lambda(N)$, which is a divisor of $\phi(N)$

group $\mathcal{K}$ (in the RSA scheme $\mathcal{K}$ is $\mathbf{Z}_{\phi(N)}$). They adapted the Shamir secret sharing scheme [18], by defining an extension ring $\mathbf{Z}[u]$, where $\mathbf{Z}[u] \cong \mathbf{Z}[x]/m(x)$, $m(x)$ is the cyclotomic polynomial $\frac{x^q-1}{x-1}$ and where q is a prime greater than $l+1$, and established that $\mathcal{K}^{q-1}$ is a module over $\mathbf{Z}[u]$.

There are two concerns with this application. In order to circumvent revealing the exponent³ $e_{\mathcal{K}}$ they have used the group $\mathcal{K}^{q-1}$ which is a module over $\mathbf{Z}[u]$. Thus shares are to be drawn from the group $\mathcal{K}^{q-1}$, yet the secret actually belongs to the group $\mathcal{K}$. (By Bertrand's postulate [15], there exists a q satisfying $l \leq q-1 \leq 2l$.) So when l is large, the length of the shares can be a burden on the shareholder (a burden in the sense of time and resources). In [10], it was established the size of shares could be halved. Here we address how to reduce the amount of time needed to perform computations.

2 Background on Desmedt Frankel Scheme

Let us discuss the threshold scheme described in [9]. $\mathcal{K}$ represents a finite abelian group. The secret will be denoted by $k \in \mathcal{K}$. A prime q is chosen such that $q \geq l+1$. (We can assume, that $O(q) = O(l)$.) Let u represent a root of the cyclotomic polynomial

$$m(x) = \sum_{j=0}^{q-1} x^j.$$

Much of the work is performed in the abelian group $\mathbf{Z}[u] \cong \mathbf{Z}[x]/m(x)$. An important observation is that for each i ($1 \leq i \leq q-1$) $\alpha_i = \sum_{j=0}^{i-1} u^j$ is a unit (an invertible element in $\mathbf{Z}[u]$) and that $\alpha_i - \alpha_j$ is a unit for all i, j , with $1 \leq i, j \leq q-1$, $i \neq j$.

Consider the group $\mathcal{K}^{q-1}$ where $\mathcal{K}^{q-1} = \mathcal{K} \times \mathcal{K} \times \cdots \times \mathcal{K}$. If $\mathbf{x} \in \mathcal{K}^{q-1}$ then $\mathbf{x} = [x_0, x_1, \dots, x_{q-2}]$. For all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{K}^{q-1}$

$$\mathbf{x}_1 + \mathbf{x}_2 = [x_{1,0} + x_{2,0}, x_{1,1} + x_{2,1}, \dots, x_{1,q-2} + x_{2,q-2}].$$

$\mathbf{0} = [0, 0, \dots, 0]$, where 0 denotes the identity in $\mathcal{K}$. For all $b \in \mathbf{Z}$, $b\mathbf{k} = [bk_0, bk_1, \dots, bk_{q-2}]$, where bk_i represents the element in $\mathcal{K}$, formed by applying k_i to itself b times. For $u \in \mathbf{Z}[u]$, $u\mathbf{k} = [0, k_0, k_1, \dots, k_{q-3}] + [-k_{q-2}, \dots, -k_{q-2}] = [-k_{q-2}, -k_{q-2} + k_0, \dots, -k_{q-2} + k_{q-3}]$. Then $u^{i+1}\mathbf{k} = u(u^{i+1}\mathbf{k})$. For all polynomials f in u with integer coefficients, $f(u) = b_0 + b_1u + \cdots + b_mu^m$, $f(u)\mathbf{k}$ is defined by $f(u)\mathbf{k} = \sum_{i=0}^m b_i(u^i\mathbf{k})$. Then $\mathcal{K}^{q-1}$ is a module over $\mathbf{Z}[u]$ (see [1]).

³ The exponent of a finite abelian group G is the smallest natural number n such that $g^n = 1$ for all $g \in G$. The Carmichael function $\lambda(N)$ is the exponent of $\mathcal{K} = \mathbf{Z}_{\phi(N)}$.

2.1 How Shares Are Computed

Given secret k , represent $\mathbf{k}$ by $[k, 0, \dots, 0] \in \mathcal{K}^{q-1}$. Each shareholder P_i ($i = 1, \dots, l$) is given share $\mathbf{s}_i$ in the following manner: Shares $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{t-1}$ are chosen uniformly randomly from $\mathcal{K}^{q-1}$. For $i = t, \dots, l$, $\mathbf{s}_i$ is defined by:

$$\mathbf{s}_i = y_{i,C_i}^{-1} \cdot \left(\mathbf{k} - \sum_{\substack{j \neq i \\ j \in C_i}} y_{j,C_i} \cdot \mathbf{s}_j \right) \quad (1)$$

where $C_i = \{1, 2, \dots, t-1, i\}$ and for $j \in C_i$

$$y_{j,C_i} = \frac{\prod_{\substack{h \in C_i \\ h \neq j}} (0 - \alpha_h)}{\prod_{\substack{h \in C_i \\ h \neq j}} (\alpha_j - \alpha_h)} \quad (2)$$

Desmedt and Frankel later observed an alternate way to define the shares. Randomly select $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{t-1} \in \mathcal{K}^{q-1}$, then the shares are defined by

$$\begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_l \end{bmatrix} = \begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{t-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{t-1} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha_l & \cdots & \alpha_l^{t-1} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_{t-1} \end{bmatrix} \quad (3)$$

Therefore for each i , $\mathbf{s}_i = \mathbf{k} + \alpha_i \cdot \mathbf{c}_1 + \cdots + \alpha_i^{t-1} \mathbf{c}_{t-1}$. Hence $\mathbf{s}_i = g(\alpha_i)$ where $g(x) = \mathbf{k} + x \cdot \mathbf{c}_1 + \cdots + x^{t-1} \mathbf{c}_{t-1}$.

2.2 How the Secret k Is Computed

When a set B of t participants wish to compute $k \in \mathcal{K}$, they can determine $\mathbf{k}$, of which the first component of $\mathbf{k}$, is the secret k . The participants determine $\mathbf{k}$ by

$$\mathbf{k} = \sum_{i \in B} y_{i,B} \cdot \mathbf{s}_i \quad (4)$$

where $y_{i,B}$ is defined by (2). We will use the F_0 to denote the function which maps any $q-1$ tuple to its first coordinate. Thus $k = F_0(\mathbf{k})$.

2.3 How Much Time Is Needed to Perform the Necessary Algebraic Operations

One of our concerns is the amount of time each shareholder uses to perform algebraic operations, and another is the amount of time the distributor needs to perform algebraic operations. As described by equations (1) and (4), a required

computation is $y_{i,B}$. (Where in the case of DISTRIBUTOR, they need to calculate such a $y_{i,B}$ “ $t(l-t)$ ” times.) There are two approaches to perform the needed calculation.

The first approach is to compute $y_{i,B} \cdot s_i$, by

$$y_{i,B} \cdot s_i = \prod_{\substack{h \in C_i \\ h \neq j}} (0 - \alpha_h) \left(\prod_{\substack{h \in C_i \\ h \neq j}} \left(\frac{1}{\alpha_j - \alpha_h} \cdot s_i \right) \right).$$

The running time to compute $y_{i,B} \cdot s_i$ in this manner, as stated by [9] is:

Theorem 1. [9] *SHAREHOLDER: performs $O(tl^2)$ group operations, and $O(l)$ inverses. In addition to the time to choose randomly $t-1$ shares. DISTRIBUTOR performs $O(t^2l^2(l-t))$ group operations and $O(tl(l-t))$ inverse operations.*

A different method is suggested if group operation is slower than integer multiplication. Instead of performing a series of group operations, a series of $\mathbf{Z}[u]$ operations are performed, until $y_{i,B}$ is formed, then one group operation is performed. [9] established the following:

Theorem 2. [9] *Each SHAREHOLDER performs $O(lt \log l)$ group operations, $O(l)$ inverses, and $O(t^3l^2(\log l)^2)$ elementary integer operations. In addition to the time needed to choose $t-1$ random shares, the DISTRIBUTOR performs $O(t^2l(l-t) \log l)$ group operations, $O(tl(l-t))$ inverse operations, and $O(t^4l^2(l-t)(\log l)^2)$ elementary integer operations.*

The goal of this paper is to devise a scheme which improves the time constraints placed on the shareholder and distributor when one uses this second method of computation as described by Theorem 2. In this case, the SHAREHOLDER performs $O(t^3l^2(\log_2 l)^2)$ integer operations, In light of the fact that $O(l) = O(q)$, we can express this as $O(t^3q^2(\log_2 q)^2)$ and $O(qt \log_2 q)$ respectively. The DISTRIBUTOR performs $O(t^4q^2(q-t)(\log_2 q)^2)$ integer operations, and $O(qt^2(q-t) \log q)$ group operations.

3 An Algorithm which Multiplies a f by an α_x

We will express the running time ⁴ in the number of elementary (“bit”) operations. From [3], given polynomials $f, g \in \mathbf{Z}[u]$, the amount of time to multiply $f \cdot g$ is $O(q^2 \log_2 C_f \log_2 C_g)$, where $\log_2 C_f$ and $\log_2 C_g$ represents the size (in bits) of the largest coefficient of f and g , respectively. This is, of course, uses a method which does not take advantage of the type of polynomial f and/or g may be. (For example if one of them is a cyclotomic polynomial α_x then we can apply the following Algorithm 31).

⁴ In order to make fair comparisons, we do want to express running time using the same metric, “number of elementary operations”.

Let $f(u) \in \mathbf{Z}[u]$, then $f(u) = a_0 + a_1u + \cdots + a_{q-1}u^{q-1}$, so we can represent f by $(a_0, a_1, \dots, a_{q-1})$. (Observe that it is possible to represent $f(u)$ as a polynomial of degree at most $q-2$, because $u^{q-1} = -u^{q-2} + \cdots - u + -1$, since $u^{q-1} + u^{q-2} + \cdots + u + 1 = 0$ in $\mathbf{Z}[u]$. However, in this paper we will not use this representation.) The following algorithm is used to multiply $\alpha_x \cdot f(u)$. Here $\deg(f)$ will denote the degree of the polynomial $f(u)$. **(In all algorithms, all subscripts are performed modulo q .)**

Algorithm 31 To multiply $\alpha_x \cdot f(u) = b_0 + b_1u + \cdots + b_{q-1}u^{q-1}$

- (0) Determine the $\deg(f)$
- (1) initialize $b_i = 0 \ \forall i$
- (2) $Temp = 0$
- (3) for $i = 0$ to $x - 1$
- (4) $b_i = a_i + Temp$
- (5) $Temp = b_i$
- (6) for $i = x$ to $\deg(f) + x - 1$
- (7) if $i \leq q - 1$
- (8) $TEMP = TEMP + a_i - a_{i-x}$
- (9) else
- (10) $TEMP = TEMP - a_{i-x}$
- (11) $b_i = b_i + TEMP$
- (12) output $(b_0, \dots, b_{q-1})$

Theorem 3. On input of α_x and polynomial $f(u)$, Algorithm 31 will correctly output the coefficients of the product $\alpha_x \cdot f(u)$. The running time of Algorithm 1 is: $O(q + \log C_f(\deg(f) + x - 1))$ where $\log C_f$ represents the number of bits needed to represent the largest coefficient of $f(u)$.

Observe that $O(q + \log_2 C_f(\deg(f) + x - 1)) \leq O(q \log_2 C_f)$. Note that if we multiply $\alpha_x \cdot \alpha_y$, then the largest coefficient of α_y is 1, and the resulting product has a largest coefficient of $\max\{x, y\}$ which is $\leq q$. If we multiply α_x with $(\alpha_y \cdot \alpha_z)$, then the largest coefficient of $(\alpha_y \cdot \alpha_z)$ is $\leq q$, and the resulting product has a largest coefficient $\leq q^2$. This argument continues in this manner. Thus an upper bound on the coefficients produced by multiplying r many α_i would be $\leq q^{r-1}$ would and gives us the following result:

Corollary 1. The time to multiply $\alpha_x \cdot \alpha_y$ is $O(q)$. The time to multiply α_x with $\prod_{y \in \mathcal{B}} \alpha_y$ (where $|\mathcal{B}| = r$) is $O(rq \log q)$.

4 Some Properties of the Cyclotomic Polynomials α_x

For each x , $1 \leq x \leq q-1$, $\alpha_x = 1 + u + \cdots + u^{x-1} = \frac{u^x - 1}{u - 1}$, and α_x is a unit. As noted earlier $u^{q-1} + u^{q-2} + \cdots + u + 1 = \alpha_q = 0$ in $\mathbf{Z}[u]$. Observe that

$u^q = u(u^{q-1} + \cdots + u + 1) - (u^{q-1} + \cdots + u + 1) + 1$, so $u^q = 1$. Thus $u^{-1} = u^{q-1}$. And so we see that for all x , u^x is a unit. Since u^x exists for all integers x , we can as well, extend the definition of α_x to all integers x . For all nonzero x , $\alpha_x = \frac{u^x - 1}{u - 1}$. Define $\alpha_0 = 0$. Observe then that $\alpha_{x+q} = \frac{u^{x+q} - 1}{u - 1} = \frac{u^x - 1}{u - 1}$. Hence $\alpha_{x+q} = \alpha_x$ for all x . Therefore $\alpha_i = \alpha_j$ if and only if $i \equiv j \pmod{q}$. Some other properties we will employ: $\alpha_x - \alpha_y = u^y \alpha_{x-y}$ and $\alpha_{q-x} = -u^{-x} \alpha_x$. (Other properties are displayed in a Lemma 1 in the Appendix.)

For each $x \in \mathbf{Z}_q^*$, observe that $x^{-1} \pmod{q}$ exists. To avoid confusion of α_x^{-1} with $\alpha_{x^{-1}}$ we will denote $x^{-1} \pmod{q}$ by r_x . Since $u^{q+1} = u$, it follows that for all y such that $y \equiv 1 \pmod{q}$ we have $u^y = u$. Hence $u^{xr_x} = u$.

4.1 How to Compute the Inverse of the Cyclotomic Polynomial α_x

From [3], given $\alpha_x \in \mathbf{Z}[u]$, with $x \not\equiv 0 \pmod{q}$, the amount of time to compute α_x^{-1} is $O(q^2)$. This is, of course, using a method which does not take advantage of the type of polynomial α_x is.

To compute α_x^{-1} where $\alpha_x = \frac{u^x - 1}{u - 1}$, we note the following:

$$\alpha_x^{-1} = \frac{u - 1}{u^x - 1} = \frac{u^{r_x x} - 1}{u^x - 1} = u^{(r_x - 1)x} + u^{(r_x - 2)x} + \cdots + u^x + 1.$$

The following is the algorithm to determine α_x^{-1} :

Algorithm 41 To determine $\alpha_x^{-1} = b_0 + b_1 u + \cdots + b_{q-1} u^{q-1}$, for $x \not\equiv 0 \pmod{q}$:

- (1) Initialize $b_i = 0 \ \forall i$
- (2) determine r_x (this is $x^{-1} \pmod{q}$)
- (3) for $i = 0$ to $r_x - 1$
- (4) $b_{ix} = 1$
- (5) output $(b_0, \dots, b_{q-1})$

Here b_{ix} refers to the coefficient of the term u^{ix} , where ix is reduced modulo q .

Theorem 4. On input $x \not\equiv 0 \pmod{q}$, Algorithm 41 will output α_x^{-1} in time $O(q \log q)$.

5 Some Results Concerning the Ratio of Two Cyclotomic Polynomials

For all $x, y \in \mathbf{Z}_q^*$ consider $\frac{\alpha_y}{\alpha_x}$. By observing that $y = y \cdot r_x \cdot x = (y \cdot r_x) \cdot x = k \cdot x$, where $k = y \cdot r_x$. We see that

$$\frac{\alpha_y}{\alpha_x} = \frac{\alpha_{kx}}{\alpha_x} = \frac{u^{kx} - 1}{u^x - 1} = u^{(k-1)x} + u^{(k-2)x} + \cdots + u^x + 1$$

Definition 1. Let $a \in \mathbf{Z}_q^*$. Let π_a denote the permutation of $\{1, 2, \dots, q-1\}$ such that $\pi_a(x) = ax \pmod{q}$. We extend the definition of π_a to the polynomial $f(u) = b_0 + b_1u + \dots + b_{q-1}u^{q-1}$ by:

$$\pi_a(f) = \sum_{i=0}^{q-1} b_i u^{\pi_a(i)}.$$

Observe that $\pi_a(f) = f(u^a)$.

It follows then that $\pi_a(\alpha_x) = 1 + u^a + u^{2a} + \dots + u^{(x-1)a}$. For example, since $\alpha_3 = 1 + u + u^2$ then $\pi_4(\alpha_3) = 1 + u^4 + u^8$. And so we see that $\pi_a(\alpha_x) = \frac{\alpha_{ax}}{\alpha_a}$. Some results concerning π_a are displayed in Lemma 2 in the appendix. We will use the following properties: (1) $\pi_1(f) = f$; (2) $\pi_{ab}(f) = \pi_a(\pi_b(f))$; and (3) $\pi_a(fg) = \pi_a(f)\pi_a(g)$, where $f, g \in \mathbf{Z}[u]$ and $a, b \in \mathbf{Z}_q^*$.

5.1 Generalizing Algorithm 31

Algorithm 31 can be generalized to the case of multiplying $\pi_x(\alpha_a)$ to $f(u)$ (here $\pi_x(\alpha_a) = \frac{\alpha_{ax}}{\alpha_x}$.) Again $f(u) = a_0 + a_1u + \dots + a_{q-1}u^{q-1}$. (For brevity, we state the algorithm in an informal manner)

Algorithm 51 To multiply $(u^{(a-1)x} + u^{(a-2)x} + \dots + u^x + 1)$ to $f(u) = b_0 + b_1u + \dots + b_{q-1}u^{q-1}$.

The product $\frac{\alpha_{ax}}{\alpha_x} \cdot f(u)$ is the same as $\pi_x(\alpha_a) \cdot f(u)$, which is equal to $\pi_x(\alpha_a \cdot \pi_{r_x}(f(u)))$. Thus we could multiply α_a to $\pi_{r_x}(f(u))$ using Algorithm 31 Then apply π_x to the output.

Theorem 5. Algorithm 51 will correctly output the product $u^{(a-1)x} + u^{(a-2)x} + \dots + u^x + 1$ with $f(u)$ with running time of $O(q \log q + q \log C_f)$

6 To Compute $y_{i,B}$

Let $\mathcal{B}$ represent a set of shareholders of cardinality t . To recover the secret, it is essential that each shareholder $i \in \mathcal{B}$ must calculate $y_{i,B}$.

$$y_{i,B} = \frac{\prod_{\substack{j \in \mathcal{B} \\ j \neq i}} (\alpha_j - 0)}{\prod_{\substack{j \in \mathcal{B} \\ j \neq i}} (\alpha_j - \alpha_i)} = \frac{\prod_{\substack{j \in \mathcal{B} \\ j \neq i}} \alpha_j}{\prod_{\substack{j \in \mathcal{B} \\ j \neq i}} u^i \prod_{\substack{j \in \mathcal{B} \\ j \neq i}} \alpha_{j-i}} = \frac{\prod_{\substack{j \in \mathcal{B} \\ j \neq i}} \alpha_j}{u^{(|\mathcal{B}|-1)i} \prod_{\substack{j \in \mathcal{B} \\ j \neq i}} \alpha_{j-i}} = u^{-(|\mathcal{B}|-1)i} \prod_{\substack{j \in \mathcal{B} \\ j \neq i}} \frac{\alpha_j}{\alpha_{j-i}}.$$

(We have applied, in the third fraction from the left, Lemma 1 statement (vi), which is the result that $\alpha_j - \alpha_i = u^i \alpha_{j-i}$.)

Observe that $\prod_{\substack{j \in \mathcal{B} \\ j \neq i}} \frac{\alpha_j}{\alpha_{j-i}}$ can be interpreted as a product $\prod \frac{\alpha_{ax}}{\alpha_x}$ where $x \in \{j - i : j \in \mathcal{B}, j \neq i\}$ and $a \equiv j \cdot r_{j-i} \pmod{q}$.

6.1 Time to Compute $y_{i,B}$

The time to determine $u^{-(|B|-1)i}$ is $O(q)$. For each j , $\frac{\alpha_j}{\alpha_{j-i}} = \frac{\alpha_{a(j-i)}}{\alpha_{j-i}}$. The amount of time to determine this a is $(\log q)^2$, which will be absorbed in to the time to compute $y_{i,B}$ as a sequence of multiplications using Algorithm 51. the time is equal to

$$q \log q + 2q \log q + \cdots + (t-1)q \log q = O(t^2 q \log q).$$

Therefore the total running time to compute $y_{i,B}$ is $O(t^2 q \log_2 q)$ which is equivalent to $O(t^2 \log_2 l)$.

Theorem 6. *The amount of time for a SHAREHOLDER to compute $y_{i,B}$ is $O(t^2 q \log_2 q)$ elementary operations. The amount of time for the DISTRIBUTOR to compute all required $y_{i,B}$ is $O(t^2 q (q-t) \log_2 q)$ elementary operations.*

Remark Observe that the SHAREHOLDER's computations of $y_{i,B}$ is $\frac{1}{tq \log_2 q}$ of the time given in Theorem 2. The DISTRIBUTOR's computations of $y_{i,B}$ is $\frac{1}{t^2 q \log_2 q}$ of the time given in Theorem 2. In practice, the amount of time to compute $y_{i,B}$ should be much smaller than $O(t^2 q \log_2 q)$. One reason is the upper bound on the size of the coefficients. When multiplying r many $\frac{\alpha_{ax}}{\alpha_x}$, we used $r \log_2 q$ as an upper bound. (This was the comment that established Corollary 1). In practice this bound is much larger than the actual coefficient. A second reason is that when one considers $\prod_{j \in B} \frac{\alpha_j}{\alpha_{j-i}}$ many cancellations may occur. Some of them are outright cancellations, others reducing to polynomials whose coefficients (in size) is much smaller. (This is where some of the results of Lemma 1 could play an important role in creating reductions.)

7 Computations within the Group

Let us assume that we are implementing this scheme to perform threshold RSA signatures. Then $\mathcal{K} = \mathbf{Z}_{\phi(N)}$. Suppose that m represents the message. We define $G = \{(m^{x_0}, m^{x_1}, \dots, m^{x_{q-2}}) : x_i \in \mathbf{K}\}$, then G is a multiplicative group. If we define $\mathbf{x} = [x_0, x_1, \dots, x_{q-2}]$ then we could interpret elements of G as $m^{\mathbf{x}}$. Recall the definition of $\mathbf{Z}[u]$, define a scalar operation on G by: for $b \in \mathbf{Z}$, then $b \cdot m^{\mathbf{x}} = m^{b\mathbf{x}}$ where $b\mathbf{x}$ represents the scalar operation defined on $\mathcal{K}^{q-1}$. Define $u \cdot m^{\mathbf{x}} = m^{u\mathbf{x}}$ where $u\mathbf{x}$ has been defined earlier. Analogously for all $f(u) \in \mathbf{Z}[u]$, define $f(u) \cdot m^{\mathbf{x}} = m^{f(u)\mathbf{x}}$, then G is a $\mathbf{Z}[u]$ module. So we have

$$m^{\mathbf{k}} = [y_{i_1,B}, \dots, y_{i_t,B}] \begin{bmatrix} m^{s_{i_1}} \\ \vdots \\ m^{s_{i_t}} \end{bmatrix}.$$

In addition we can reinterpret equation (3) by:

$$\begin{bmatrix} m^{s_1} \\ m^{s_2} \\ \vdots \\ m^{s_t} \end{bmatrix} = \begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{t-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{t-1} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha_t & \cdots & \alpha_t^{t-1} \end{bmatrix} \begin{bmatrix} m^k \\ m^{c_1} \\ \vdots \\ m^{c_{t-1}} \end{bmatrix}.$$

These last two equations point out that although this threshold scheme has described shares in $\mathcal{K}^{q-1}$, computations of the secret is actually done in G . Therefore it may be misleading for us to count the number of group operations in $\mathcal{K}$ (or in $\mathcal{K}^{q-1}$). For a single group operation is a multiplication in $\mathbf{Z}_N$. We use $(\log_2 N)^2$ to represent the amount of time to perform multiplication in $\mathbf{Z}_N$ (although better implementations exist).

The emphasis here is on the computations a shareholder will have to perform within the group. Recall $\mathcal{K}$ is $\mathbf{Z}_{\phi(N)}$, and in threshold RSA no participant can be given any information concerning $\phi(N)$. Shares are actually $q-1$ dimensional vectors formed from $\mathbf{Z}_{\phi(N)}^{q-1}$. From a shareholder's view this will look like a $q-1$ dimensional integer vector. Thus computations can be performed within the group using integer addition. The secret is d , where the RSA public key is (e, N) and $ed = 1 \pmod{\phi(N)}$. Therefore $\mathbf{k} = [d, 0, \dots, 0] = \sum_{i \in B} y_{i,B} \mathbf{s}_i$. If a set of t participants would like to sign a message m , then they will not send their subshares of d (i.e. they will NOT send $y_{i,B} \cdot \mathbf{s}_i$), but rather they will send partial signatures. They could send $m^{y_{i,B} \cdot \mathbf{s}_i}$. If all t participants sent $m^{y_{i,B} \cdot \mathbf{s}_i}$ then a combiner would get m^d by

$$m^d = F_0(m^{[d,0,\dots,0]}) = F_0\left(\prod_{i \in B} m^{y_{i,B} \cdot \mathbf{s}_i}\right).$$

However we must point out that this method wastes resources. That is, the combiner is actually computing $m^{[d,0,\dots,0]} = [m^d, 1, 1, \dots, 1]$, where the only element of interest is m^d . Consider the following: First, recall that F_0 is a function which maps a j -tuple to its first coordinate. Second, suppose that B is a nonempty set, and that for all $j \in B$, $\mathbf{z}_j \in \mathcal{K}^{q-1}$, $\mathbf{x}_j = [x_{j,0}, x_{j,1}, \dots, x_{j,q-2}]$.

$$F_0\left(\prod_{j \in B} m^{\mathbf{z}_j}\right) = \prod_{j \in B} F_0(m^{\mathbf{z}_j}) = m^{F_0(\sum \mathbf{z}_j)} = m^{\sum z_{j,0}} \quad (5)$$

So we see by equation (5), to compute m^d , all that is needed is $F_0(y_{i,B} \cdot \mathbf{s}_i)$.

In [11], Desmedt and Frankel used the same ring $\mathbf{Z}[u]$, and described the following alternative for shareholders to construct partial signatures.

Suppose $y_{i,B} = c_{i,0} + c_{i,1}u + \cdots + c_{i,q-1}u^{q-1}$. Before we can compute $y_{i,B} \mathbf{s}_i$, we must use the equation $u^{q-1} = -u^{q-2} + \cdots + -u + -1$ to convert $y_{i,B}$ to a polynomial of degree $q-2$. We represent $y_{i,B}$, reduced to degree $q-2$, by $a_{i,0} + a_{i,1}u + \cdots + a_{i,q-2}u^{q-2}$. Thus $a_{i,j} = c_{i,j} - c_{i,q-1}$.

The first coordinate of $y_{i,B} \cdot \mathbf{s}_i$ can be determined without computing all $q-1$ coordinates. Using our notation the first coordinate will be

$$a_{i,0}s_{i,0} + (a_{i,2} - a_{i,1})s_{i,q-2} + (a_{i,3} - a_{i,2})s_{i,q-3} + \cdots + (a_{i,q-2} - a_{i,q-3})s_{i,2} + -a_{i,q-2}s_{i,1} \quad (6)$$

Since $c_{i,j}$ is bounded between 0 and q^{t-1} , we find that $a_{i,j}$ is bounded between $-q^{t-1}$ and q^{t-1} .

Algorithm 71 [11] To compute the partial signature for participant P_i
 Let B be a set of t participants, such that $i \in B$.

- (1) Compute $y_{i,B}$.
- (2) Compute equation (6). Denote this integer by $temp_i$.
- (3) Compute $m^{temp_i} \pmod{N}$, this is the partial signature of P_i .

If P_i was to determine their partial signature in a manner in which it omitted computing $temp_i$ as an integer, and computed m^{temp_i} immediately after computing $y_{i,B}$ then the amount of time would be $O(tl \log_2 l)$ group computations. Since the group computations would be in $\mathbf{Z}_N$ using multiplication, the actual cost would be on the order $O(tl \log_2 l (\log_2 N)^2)$.

Consider the amount of time to compute the partial signature in the manner described by Algorithm 71. To compute $temp_i$: consider the computation of

$$a_{i,0}s_{i,0} + a_{i,2}s_{i,q-2} + \cdots + a_{i,q-2}s_{i,2}.$$

As we had described earlier, $-q^{t-1} \leq a_{i,j} \leq q^{t-1}$ for all j . Therefore to compute $a_{i,j}s_{i,q-j}$ will take $O(\log_2 q^t \cdot \log_2 \phi(N)) = O(t \log_2 q \cdot \log_2 \phi(N))$. Thus to compute all q of them would be $O(tq \log_2 q \cdot \log_2 \phi(N))$. (which we will express as $O(tl \log_2 l \cdot \log_2 \phi(N))$ since $O(l) = O(q)$). Reminder this cost is given in elementary operations. Equation (6) can be performed in three steps: Compute $a_{i,0}s_{i,0} + a_{i,2}s_{i,q-2} + \cdots + a_{i,q-2}s_{i,2}$, compute $a_{i,1}s_{i,q-2} + a_{i,2}s_{i,q-3} + \cdots + a_{i,q-2}s_{i,1}$ then take the difference. Thus step (2) of Algorithm 71 can be completed in $O(tl \log_2 l \cdot \log_2 \phi(N))$. Now note that $a_{i,0}s_{i,0} + a_{i,2}s_{i,q-2} + \cdots + a_{i,q-2}s_{i,2}$ is bounded above by $qq^{t-1} \cdot \phi(N) = q^t \phi(N)$. To compute m^{temp_i} , will take on the order of maximum $O((t \log_2 l + \log_2 \phi(N))(\log_2 N)^2)$. Thus the total time to compute the partial signature is $O(t \log_2 l + \log_2 \phi(N))(\log_2 N)^2 + O(tl \log_2 l \cdot \log_2 \phi(N)) + O(t^2 l \log_2 l)$. Simplifying this we see the cost of generating a signature is:

$$O(tl \log_2 l(t + \log \phi(N)) + [t \log_2 l + \log_2 \phi(N)](\log_2 N)^2). \quad (7)$$

Remark Observe that if $tl \log_2 l \leq (\log_2 N)^2$ then equation (7) reduces to $O([t \log_2 l + \log_2 \phi(N)](\log_2 N)^2)$. Now consider two cases: 1) $\log_2 \phi(N) \geq t \log_2 l$ and 2) $\log_2 \phi(N) < t \log_2 l$. In the case when $\log_2 \phi(N) \geq t \log_2 l$, this implies that time to compute a threshold RSA partial signature is equivalent to the time to compute a RSA signature. In the case when $\log_2 \phi(N) < t \log_2 l$ we see a reduction in the amount of time to perform group operations by $\frac{1}{l}$. (This reduction is for the SHAREHOLDER.)

Because the DISTRIBUTOR needs to form the entire m^x , we cannot apply Algorithm 71. However the DISTRIBUTOR is performing group operations in the additive group $\mathbf{Z}_{\phi(N)}$. The DISTRIBUTOR has to: perform $O(t^2 l(l - t) \log_2 l)$ elementary operations, generate t random vectors of length l , and $O(lt^2(l - t) \log_2 l)$ additions in $\mathbf{Z}_{\phi(N)}$. These additions have a cost of $O(lt^2(l -$

$t) \log_2 l \log_2 \phi(N)$. This latter cost refers to the creation of $(l-t)$ vectors of length l . Therefore the DISTRIBUTOR requires $O(lt^2(l-t)(\log_2 l + \log_2 l \log_2 \phi(N))) = O(lt^2(l-t) \log_2 l \log_2 \phi(N))$. Realize that this cost is a one time cost (for the duration of the key d), whereas the SHAREHOLDERS will apply their algorithm every time they need to sign a message.

Lastly the COMBINER is required to multiply t elements in $\mathbf{Z}_N$. This has a cost of $O(t(\log_2 N)^2)$.

8 Shoup's Threshold RSA Signature Scheme

Our interest in Shoup's RSA threshold signature scheme [19] is that it allows us to make a comparison between two non interactive RSA threshold signature schemes. The share size in Shoup's scheme is on the order of the size of the secret. Which is a tremendous improvement over Desmedt-Frankel's scheme. Our sole interest is share size and time required to perform computations. So we will omit comparison between the two schemes outside this realm.

$N = pq$ where p and q are primes such that $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are prime. Let $n = p'q'$. The dealer chooses an RSA exponent e . The dealer chooses $a_1, a_2, \dots, a_{t-1}$ at random from $\{0, \dots, n-1\}$. The dealer defines the polynomial $f(x) = d + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1}$. For each $i \in \{1, \dots, l\}$ the dealer sends participant the share

$$s_i = f(i) \pmod{n}.$$

Let $\Delta = l!$. For any each subset S of t points from set $\{0, 1, \dots, l\}$ and for any $i \in \{0, 1, \dots, l\} \setminus S$ and $j \in S$ define

$$\lambda_{i,j}^S = \Delta \frac{\prod_{j' \in S \setminus \{j\}} (i - j')}{\prod_{j' \in S \setminus \{j\}} (j - j')}$$

Then for all $i \in \{1, \dots, l\}$

$$\Delta f(i) \equiv \sum_{j \in S} \lambda_{i,j}^S f(j) \pmod{n}$$

Next a hash function is needed to map messages to elements of $\mathbf{Z}_N^*$. If $x = H(m)$ for message m , then a signature for m is a y such that $y^e = x$ (modulo n).

Signature shares. Each participant has received s_i , they then form $x_i = x^{2\Delta s_i}$ which belongs to the subgroup consisting of all squares in $\mathbf{Z}_N^*$.

Each member of a set S which consist of t participants will: first compute $x_{i_j}^2 = x^{4\Delta s_{i_j}}$. (These steps are separated because a proof of correctness is sent using this $x_{i_j}^2$, which is omitted from our discussion.) Secondly, the participant sends $x_{i_j}^{2\lambda_{0,i_j}^S}$ to the combiner.

The combiner computes w by:

$$w = x_{i_1}^{2\lambda_{0,i_1}^S} \cdots x_{i_t}^{2\lambda_{0,i_t}^S}.$$

It follows then that $w^e = x^{e'}$ where $e' = 4\Delta^2$. Since $\gcd(e', e) = 1$, by using the extended Euclidean Algorithm one can find a and b such that $ae' + be = 1$. The signature is $y = w^a x^b$. The signature can be verified by $y^e = x$

8.1 The Running Time of Shoup’s Threshold Scheme

Our interest is to make a comparison between the running time of Shoup’s scheme and our method of computation for the scheme [9]. We point out the computation of x_{i_j} can be described as follows: $x_{i_j} = (message)^{2\Delta s_i}$. The size of $2\Delta s_i$ is on the order of $O(2l!\phi(N))$. Hence this will take $O(l\log_2 l + \log_2 \phi(N))$ group operations. Thus the computation is on the order of $O((l\log_2 l + \log_2 \phi(N))(\log_2 N)^2)$.

The computation of $x_{i_j}^{2\lambda_{0,i_j}^S}$ adds to this, a cost of the same size. Thus we see the computation cost of Shoup’s scheme is $O((l\log_2 l + \log_2 \phi(N))(\log_2 N)^2)$. Observe that we have established that the cost of group operations within [9] using our Algorithm 71 is $O(t\log_2 l(\log_2 N)^2 + \log_2 \phi(N)(\log_2 N)^2)$. Hence whenever $t\log_2 l > \log_2 \phi(N)$ we see the threshold scheme in [9] requires less computations than Shoup’s scheme. And in the case $t\log_2 l \leq \log_2 \phi(N)$, we see that the computational cost for computing partial signatures in both schemes is equivalent to computing a RSA signature. (This argument is assuming $lt\log_2 l < (\log_2 N)^2$.)

In Shoup’s scheme The DISTRIBUTOR is required to compute $f(i)$, which requires a time of $O(t(\log_2 \phi(N))^2)$. The COMBINER is required to multiply t elements in $\mathbf{Z}_N$ and to compute w^a and x^b . This is a cost of $(t + \log_2 a + \log_2 b)(\log_2 N)^2$.

	Shoup’s scheme	Desmedt-Frankel scheme
Size of share	1	l
SHAREHOLDER (time)	$(l\log_2 l + \log_2 \phi(N))(\log_2 N)^2$	$tl\log_2 l(t + \log \phi(N)) + [t\log_2 l + \log_2 \phi(N)](\log_2 N)^2$.
DISTRIBUTOR (time)	$t(\log_2 \phi(N))^2$	$lt^2(l - t)\log_2 l\log_2 \phi(N)$
COMBINER (time)	$(t + \log_2 a + \log_2 b)(\log_2 N)^2$	$t(\log_2 N)^2$

Within this table, we have omitted the computation cost required by the DISTRIBUTOR for producing random elements. All times are given using order notation.

9 Summary and Open Problems

In this paper we have described algorithms which effectively speed up computations in the threshold RSA scheme developed by [9]. Our algorithms have addressed how to speed up

computations within $\mathbf{Z}[u]$ and group operations. We point out that the computations within $\mathbf{Z}[u]$ are significantly less when using the algorithms that we have discussed. We also point out that our estimate of running time is affected by our bound on $y_{i,B}$. An upper bound on the coefficients in $y_{i,B}$ is affected by the indices of the α 's used to compute $y_{i,B}$. For example, if set B is such that all $j \in B$ are near i then coefficients in $y_{i,B}$ will be smaller. Also, within Lemma 1 there exists an equation $\alpha_{q-x} = -u^x \alpha_x$, this implies that one may assume that all indices of α 's are $\leq \frac{q+1}{2}$. Further, it should be noted that in [10], they have shown that share size can be halved. (Both of these ideas would reduce computational time, but in terms of order notation the time would not change.)

We point out that to use the threshold scheme of Desmedt-Frankel, one is generating larger shares. However, we have seen computational time may be smaller than threshold RSA with share size proportional to the secret. Note that in terms of resources, communication between the SHAREHOLDERS and the combiner will be proportional to the secret. That is, the SHAREHOLDER will have to compute with this larger share, but the partial signatures they generate will not be of this size. The only time when communication will involve the use of these larger shares is during the process of dealing out the shares.

Possible future work. Is it possible to reduce the number of computations the DISTRIBUTOR makes? Is it possible to create better bounds on the size of the coefficients of $y_{i,B}$? Also, possible work could be done on reducing the length of the share. The reason the share is of length $q \geq l + 1$, is that one needs l units (i.e. α_i). However, these units need to be special in the sense that their difference is also a unit. (The reason is that the security of this scheme is based on the Vandermonde matrix.) If one can find a smaller q for which l units exist with this property, we would see an immediate improvement in both share size and computational cost. (This is referred to as the *Lenstra constant*.)

References

1. W. Adkins and S. Weintraub. "Algebra, an approach via module theory". Springer-Verlag, NY, 1992.
2. A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, 1974.
3. E. Bach and J. Shallit, *Algorithmic Number Theory*, MIT Press, 1996
4. C. Boyd, *Digital Multisignatures*, Cryptography and coding, Clarendon Press, 1989, pp 241-246.
5. H. Cohen. "A course in Computational Algebraic Number Theory". Springer-Verlag, NY, 1996.
6. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. "How to share a function". In *Proceedings of the twenty-sixth annual ACM Symp. Theory of Computing (STOC)*, pages 522-533, 1994.
7. Y. Desmedt and Y. Frankel. *Threshold Cryptosystems* In Advances of Cryptology-Crypto '89, pages 307-315, 1989.
8. Y. Desmedt and Y. Frankel. *Shared generation of authenticators and signatures*. In Advances of Cryptology- Crypto '91, 1991.
9. Y. Desmedt and Y. Frankel, *Homomorphic Zero Knowledge Threshold schemes over any finite Abelian Group* Siam J. Disc. Math, vol. 7(4), pp 667-679.
10. Y. Desmedt, B. King, W. Kishimoto, and K. Kurosawa, *A comment on the efficiency of secret sharing scheme over any finite abelian group*, Information Security and Privacy, ACISP '98, Lecture Notes in Computer Science 1438, 1998, pp 391-402.

11. Y. Frankel and Y. Desmedt. *Parallel reliable threshold multisignature*. Tech. report TR-92-04-02, Univ. of Wisconsin-Milwaukee, 1992.
12. Y. Frankel, P. Gemmel, P. Mackenzie, and M. Yung. *Proactive RSA* In Advances of Cryptology- Crypto '97, 1997 Lecture Notes in Computer Science 1294, Springer Verlag, 1997, pp 440-454.
13. Y. Frankel, P. Gemmel, P. Mackenzie, and M. Yung. *Optimal-Resilience Proactive Public-key cryptosystems* In Proc. 38th FOCS, pp 384-393, IEEE, 1997
14. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. *Robust and efficient sharing of RSA functions*. In Advances of Cryptology- Crypto '96, Lecture Notes in Computer Science 1109, Springer Verlag, 1996, pp 157-172.
15. H.L. Keng, *Introduction to Number Theory*, Springer Verlag, NY, 1982.
16. T. Rabin. *A Simplified Approach to threshold and proactive RSA*. In Advances of Cryptology- Crypto '98, 1998
17. R. Rivest, A. Shamir, and L. Adelman, *A method for obtaining digital signatures and public key cryptosystems*, Comm. ACM, 21(1978), pp 294-299.
18. A. Shamir, *How to share a secret*, Comm. ACM, 22(1979), pp 612-613.
19. V. Shoup. "Practical Threshold Signatures" Eurocrypt'2000, <http://philby.ucsd.edu/cryptolib/1999.html>

10 Appendix

Lemma 1. (i) $u^q = 1$. (ii) $\alpha_1 = 1$. (iii) $\alpha_0 = 0$. (iv) $\alpha_{x+q} = \alpha_x$. (v) $u\alpha_x + 1 = \alpha_{x+1}$. (vi) $\alpha_x - \alpha_y = u^y\alpha_{x-y}$. (vii) $\alpha_x + u^x\alpha_{q-x} = 0$. (viii) $\alpha_{q-x} = -u^{-x}\alpha_x$. (ix) $\alpha_{-x} = \alpha_{q-x}$.

Lemma 2. let $c \in \mathbf{Z}$ and $f, g \in \mathbf{Z}[u]$, $a, b \in \mathbf{Z}_q^*$ then we have the following (i) $\pi_1(f(u)) = f(u)$; (ii) $\pi_a(cf(u)) = c\pi_a(f(u))$; (iii) $\pi_a(f(u) + g(u)) = \pi_a(f(u)) + \pi_a(g(u))$; (iv) $\pi_a(f(u) \cdot g(u)) = \pi_a(f(u)) \cdot \pi_a(g(u))$; (v) $\pi_{ab}(f(u)) = \pi_a(\pi_b(f(u))) = \pi_b(\pi_a(f(u)))$; (vi) $\frac{\alpha_{kx}}{\alpha_x} = \pi_x(\alpha_k)$; (vii) $\pi_a\left(\frac{f(u)}{g(u)}\right) = \frac{\pi_a(f(u))}{\pi_a(g(u))}$; and (viii) $\frac{1}{\pi_a(\alpha_x)} = \pi_a(\alpha_x^{-1})$

Sharing Block Ciphers^{*}

Ernie Brickell¹, Giovanni Di Crescenzo², and Yair Frankel³

¹ Intel Corporation, USA.

² Telcordia Technologies, Inc., 445 South Street, Morristown, NJ, 07960, USA.

giovanni@research.telcordia.com

³ eCash Technologies, Inc., USA.

yfrankel@cryptographers.com

Abstract. Threshold cryptosystems use algebraic properties such as homomorphisms in order to allow several parties to jointly evaluate a cryptographic primitive. Several cryptographic primitives, however, avoid -by definition- the use of algebraic properties, or otherwise their security is compromised; this is the case, for instance, of block ciphers, pseudo-random functions, and pseudo-random permutations. Is it then impossible to construct a threshold cryptosystem in order to share the computation of a block cipher ?

In this paper we get around this apparent impossibility by sharing the computation of the *composition* of block ciphers. Although less efficient than a single cipher, a composition of block ciphers is still significantly more efficient than a public-key encryption. We show that the problem of sharing the composition of block ciphers is tightly related to a variant of secret sharing, which we call *sequence sharing*. We define and investigate sequence sharing schemes by presenting matching upper and lower bounds for several access structures of special interest. These schemes imply constructions of block cipher sharing schemes for the same access structures, each having optimal (under this approach) number of compositions.

1 Introduction

The power of sharing computation in a cryptosystem is crucial in several real-life applications of Cryptography. Threshold cryptosystems have played an important role in the study of secure distributed computation. Cryptographic primitives and tasks to which threshold cryptosystems have been applied include several variants of digital signature, identification, public-key encryption and authentication schemes. It is of great interest to extend the domain of cryptographic tasks to which threshold cryptography can be applied. In this paper we consider new primitives, such as block ciphers, which are typically used for tasks as private-key encryption and private-key authentication. We overcome their apparent elusiveness to threshold cryptography by proposing and investigating a

^{*} Copyright © 2000, Telcordia Technologies, Inc. All Rights Reserved.

methodology for their secure usage in distributed computation. The methodology automatically extends to pseudo-random permutations, and pseudo-random functions, which are used for several other tasks in Cryptography as well.

The homomorphism-based paradigm. Since their introduction [7,6], threshold cryptography schemes have relied their constructions on the following basic paradigm. A function, of different nature according to the specific cryptographic task, has to be computed by several parties. The function enjoys desirable algebraic properties, topically homomorphisms, which allow to divide it into separately computable pieces. Each party can then compute a single piece and then all computations can be combined to result in a final computation of the original function. The validity of the combination relies in a crucial way on the algebraic properties of the function.

Apparently non-algebraic primitives. Despite the success of this paradigm, there are still cryptographic primitives to which threshold cryptography has not been applied yet. This is the case, for instance, of block ciphers, as well as pseudo-random permutations and pseudo-random functions. In fact, it is not clear a priori that it is possible to apply threshold cryptography to such primitives. One reason being that a homomorphic function or permutation can easily seem to be insecure as a block cipher or a pseudo-random function. Indeed, a rule of thumb in the design of block ciphers is that of avoiding algebraic relations of any sort between inputs and outputs.

Results on block ciphers: a new methodology. We present a formal definition of block cipher sharing schemes, where security against unqualified sets of participants is stated in the sense of impossibility of predicting the output of the shared cipher. More precisely, we consider two formalizations of security. The first one, called indistinguishability, follows the approach envisioned by Shannon, of requiring block ciphers to be indistinguishable from ideal (namely, random) ciphers. The second one is new and particularly suits our study case of threshold cryptography; we call it unpredictability since, informally, it requires the output of a block ciphers (using an unknown key) to be hard to predict on any input. We relate the two formalizations, and express the security of block cipher sharing schemes according to the second. We then propose the following methodology for applying threshold cryptography to block ciphers: rather than sharing a block cipher, we would like to share the *composition* of a block cipher. Namely, first we iterate the block cipher a sufficiently large number of times, and then we ask each participant to compute only some of the iterations. Specifically, we share among n participants a function of the form $F_k(x) = f_{k_m}(\dots f_{k_2}(f_{k_1}(x)))$, where $k = (k_1, \dots, k_m)$, $m \in \text{poly}(n)$ and f is itself a secure cipher. Then we can securely share F_k by sharing the sequence $(k_1, \dots, k_m)$ among the n participants. The security of the composed cipher F will be at least as strong as the atomic cipher f . Multiple repetitions of atomic ciphers are certainly less efficient than computing a single cipher; however, we note that private key ciphers are often times orders of magnitude faster than their public key counterparts. In essence, even a large number of repeated applications of a block cipher would be faster than computing an RSA function.

Results on secret sharing: a new variant. The suggested methodology gives rise to a new problem, of independent interest, which can be considered as a variant of secret sharing ([4,10]). We recall that a secret sharing scheme is a technique to distribute a secret (key) k into shares among a set of n participants in such a way that only qualified subsets, pooling their information together, can reconstruct the secret k , but subsets of participants that are not enabled to recover the secret obtain no information on the secret (see also [5,11,3,14]). In this paper we consider a variant which we call *sequence sharing*. Here, a sequence $k = (k_1, \dots, k_m)$, where k_i is in some alphabet K_i and $m \in \text{poly}(n)$, is shared by providing shareholders with the “secret components” k_i such that only a quorum of shareholders has possession of the whole sequence yet less than a quorum does not have possession of least one of the k_i . We present a formal definition of sequence sharing schemes. We show that a sequence sharing scheme for a certain access structure can be used to transform a block cipher into a block cipher sharing scheme. Then we present lower bounds on the length of the sequence to be shared for three access structures of interest: threshold structures, OR-of-ANDs structures, and graph-based structures. Our lower bounds are general and potentially apply to several other classes of access structures. For the same specific access structures, we construct sequence sharing schemes that achieve matching upper bounds, thus automatically providing block cipher sharing schemes of optimal (with respect to this approach) sequence size.

Organization of the paper. In Section 3 we review the definition of secret sharing schemes and present the definition of sequence sharing schemes. In Section 2 we review the definition of block ciphers and define their security both in the indistinguishability and unpredictability sense, and present the definition of block cipher sharing schemes. In Section 4 we show the connection between sequence sharing and block cipher sharing. In Section 5 we present lower bounds for sequence sharing schemes and applications of the lower bound technique to specific access structures. Upper bounds for sequence sharing schemes for such access structures are given in Section 6.

2 Definitions on Block Ciphers

We start by reviewing some basic definitions about block ciphers. Then we define and discuss two security notions for block ciphers: a first one, often considered in the literature, which we call *indistinguishability*, following the intuition that a secure cipher should be indistinguishable from a random cipher; and a second one, that seems to be new, which we call *unpredictability*, following the intuition that the output of a secure cipher on any input should be unpredictable. Finally we present and formally define the new notion of block ciphers sharing schemes.

Block ciphers: basic definitions. Block ciphers, first envisioned by Shannon [12], are a basic cryptographic primitive, and are today considered the most efficient tool for private-key encryption. Formally, let κ, l be positive integers; a function F from $\{0, 1\}^\kappa \times \{0, 1\}^l$ to $\{0, 1\}^l$ is a *block cipher* if for each $k \in \{0, 1\}^\kappa$,

the function $F(k, \cdot)$ is a permutation. Intuitively, one can think of F as a 2^κ by 2^l table, with entry (k, x) containing $F(k, x)$ and where each row is a permutation of $\{0, 1\}^l$. For convenience, we also define $F_k : \{0, 1\}^l \rightarrow \{0, 1\}^l$, for each $k \in \{0, 1\}^\kappa$, as $F_k(x) = F(k, x)$, namely, the permutation of the k th row of the table. Here, by l we denote the *block length* of the cipher, by κ the *key length* of the cipher, and by $BC(\kappa, l)$ the set of all block ciphers with key length κ and block length l . Although the function F does not have an inverse function, it does have a well defined inverse block cipher. When it is clear from context that F is a block cipher then we will let F^{-1} denote the *inverse block cipher* of F from $\{0, 1\}^\kappa \times \{0, 1\}^l$ to $\{0, 1\}^l$, which is defined as follows: $F^{-1}(k, y) = F_k^{-1}(y)$. That is, $F^{-1}(k, y) = x$ if and only if $F(k, x) = y$. We note that a block cipher can also be considered as a finite version of a pseudo-random permutation.

Block ciphers: security model. Let $ACset = \{F, G, H, \dots\}$ be a set of block ciphers in $BC(\kappa, l)$. We will be interested in operators which map block ciphers in $ACset$ to a block cipher in $BC(\kappa', l')$, for some values κ', l' . As an example, consider the following operator, consisting of the cascaded composition of ciphers F, G, H , each with an independently chosen key. Formally, define operator $\text{Triple}[ACset] : BC(\kappa, l)^3 \rightarrow BC(3\kappa, l)$ as

$$\text{Triple}[ACset]((k_1, k_2, k_3), x) = F(k_1, G(k_2, H(k_3, x)))$$

for all $k_1, k_2, k_3 \in \{0, 1\}^\kappa$ and all $x \in \{0, 1\}^n$.

First security notion: indistinguishability. Let P_l be the set of permutations over $\{0, 1\}^l$. We say that a cipher C is a *random cipher* if C is uniformly and independently chosen from P_l . The security of each operator in the indistinguishability sense will be parameterized by the probability that an adversary can distinguish a random cipher from the cipher obtained by instantiating the operator with atomic random ciphers. In this setting there is a set $ACset$ of atomic ciphers $F, G, H, \dots$, and an additional cipher O , and the adversary is a computationally unlimited algorithm, making possibly probabilistic internal computation and possibly adaptive queries to such oracles. The adversary is allowed to make queries of two types. The queries of type 1 can be written as (C, k, x) , where $k \in \{0, 1\}^\kappa$, $x \in \{0, 1\}^l$, and either C or $C^{-1} \in ACset$, and are answered with a value $y \in \{0, 1\}^l$, meaning that $C(k, x) = y$. The queries of type 2 can be written as (C, x) , where $x \in \{0, 1\}^l$, and either $C = O$ or $C = O^{-1}$, and are answered with a value $y \in \{0, 1\}^l$, meaning that $C(x) = y$. We will refer to such computation of the oracle as $A^{O, ACset}$. The cipher O is randomly chosen to be either a random cipher or a cipher obtained by some composition operator, call it Op , over the ciphers in $ACset$. The goal of the adversary is to distinguish which of these two cases holds. Denoting by t the number of queries of type 1, and by q the number of queries of type 2 made by the adversary, we measure the success of the adversary as the difference of the two probabilities

$$\begin{aligned} P_A^0(Op, \kappa, l, q, t) &= \text{Prob} [F, G, H, \dots \leftarrow BC(\kappa, l); ACset \leftarrow \{F, G, H, \dots\}; \\ &\quad O \leftarrow Op[ACset] : A^{O, ACset} = 1]; \\ P_A^1(\kappa, l, q, t) &= \text{Prob} [F, G, H, \dots \leftarrow BC(\kappa, l); ACset \leftarrow \{F, G, H, \dots\}; \end{aligned}$$

$$O \leftarrow P_l : A^{O, ACset} = 1 \text{].}$$

Specifically, for all adversaries A and any κ, l, q, t , we define

$$\text{DIST}_A(Op, \kappa, l, q, t) = P_A^0(Op, \kappa, l, q, t) - P_A^1(\kappa, l, q, t).$$

The goal of the analysis of the security of a cipher obtained from a certain operator is to bound such quantity as a function of κ, l, q, t . The smaller such quantity the more secure the cipher is. We say that an operator Op is (t, q, ϵ) -*indistinguishable* if for all adversaries A it holds that $\text{DIST}_A(Op, \kappa, l, q, t) \leq \epsilon$.

Second security notion: unpredictability. The security of each operator in the unpredictability sense will be parameterized by the probability that an adversary can compute a valid input-output pair of the cipher obtained by instantiating the operator with an atomic ideal cipher. In this setting there is a set $ACset$ of atomic ciphers $F, G, H, \dots$, and the adversary is a computationally unlimited algorithm, making possibly probabilistic internal computation and possibly adaptive queries to such oracles. He is allowed to make queries of the type (C, k, x) , where $k \in \{0, 1\}^\kappa$, $x \in \{0, 1\}^l$, and either C or $C^{-1} \in ACset$, which are answered with a y , where $y \in \{0, 1\}^l$, meaning that $C(k, x) = y$. We will refer to such computation of the oracle as A^{ACset} . The goal of the adversary is to output a pair (x, y) that is a valid input-output pair for cipher $Op[ACset]$. Specifically, define probability $\text{PRED}_A(Op, \kappa, l, t)$ as equal to

$$\begin{aligned} &\text{Prob} [F, G, H, \dots \leftarrow BC(\kappa, l); ACset \leftarrow \{F, G, H, \dots\}; \\ &O \leftarrow Op[ACset]; (x, y) \leftarrow A^{ACset} : O(x) = y \text{].} \end{aligned}$$

We say that an operator Op is (t, ϵ) -*unpredictable* if for any adversary A , it holds that $\text{PRED}_A(Op, \kappa, l, t) \leq 2^{-l} + \epsilon$.

Relationship between the two security notions. The concepts of indistinguishability and unpredictability for block ciphers are related in the sense that the former is a less stringent requirement than the second. This is formally expressed in the following

Lemma 1. *Let $ACset$ be a set of atomic ciphers and let Op be an operator. For any adversary A there exists an adversary A' such that for all κ, l, t , it holds that*

$$\text{PRED}_A(Op, \kappa, l, t) = \text{DIST}_{A'}(Op, \kappa, l, 1, t).$$

Proof. We show that any adversary breaking Op in the unpredictability sense can be turned into an adversary breaking Op in the indistinguishability sense that makes only one query to the oracle O . Specifically, given an adversary A in the unpredictability setting, let $\text{PRED}_A(Op, \kappa, l, t)$ be its associated probability of returning an input-output pair for Op . Then consider adversary A' doing the following in the indistinguishability setting: A' runs algorithm A and sets (x, y) as the input-output pair returned by A ; then A' asks query x to oracle O ; if O replies with $y = O(x)$ then A' returns: 1 else A' returns: 0. Now, notice that for the given

adversary A' , it holds that: 1) $P_{A'}^1(Op, \kappa, l, 1, t) = 2^{-l}$, since in the experiment of $P_{A'}^1$ the oracle O is a random cipher; 2) $P_{A'}^0(Op, \kappa, l, 1, t) = \text{PRED}_A(Op, \kappa, l, t)$, because of the equivalence between the experiments underlying such two probabilities. The lemma follows. $\square$

Block cipher sharing schemes. Informally, a block cipher sharing scheme is a method for distributing the computation of a block cipher among a set of participants. Specifically, some information is distributed to participants in such a way that all ‘qualified’ subsets of participants are able to evaluate the block cipher over any input, while all ‘unqualified’ subsets of participants can evaluate the block cipher not significantly better than guessing at random the output or the unknown key of the cipher (this is formalized using the unpredictability notion introduced before). As before, the set of qualified subsets of participants is specified by a monotone access structure. We consider a model including n participants who interact among them in order to produce a final computation and send it to the intended receiver of this result. In particular, no combiner algorithm is required. We now present a formal definition of block cipher sharing schemes.

Definition 1. Let n, t, q, s be positive integers, let $\epsilon \geq 0$, let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of participants, let $\mathcal{A}$ be a monotone access structure over set $\mathcal{P}$, let $ACset = \{F, G, H, \dots\}$ be a set of random ciphers, and let $Op[ACset] : BC(\kappa, l)^s \rightarrow BC(\kappa, l)$ be an operator. A *block cipher sharing scheme* for cipher $Op[F]$ and access structure $\mathcal{A}$ is a pair $(\mathcal{K}, \mathcal{E})$, where $\mathcal{K}$ is the *key-sharing algorithm*, and $\mathcal{E}$ is the *evaluation protocol*. Algorithm $\mathcal{K}$ is run by a dealer; on input a string k , and some random string r , $\mathcal{K}$ returns an n -tuple $(s_1, \dots, s_n)$. Protocol $\mathcal{E}$ is run by a subset of participants. At the beginning of the protocol each participant P_i has as input s_i and a string $x \in \{0, 1\}^l$. At the end of the protocol the value $y = F(k, x)$ is computed by this subset of participants (and implicitly sent to the intended receiver) or a failure symbol $\perp$ is returned.

We say that pair $(\mathcal{K}, \mathcal{E})$ is a *block cipher sharing scheme* for cipher $Op[ACset]$ and access structure $\mathcal{A}$ if it satisfies the properties of Correctness and Unpredictability, as defined below.

1. *Correctness.* For any $A = \{P_{i_1}, \dots, P_{i_u}\} \in \mathcal{A}$, any $k \in \{0, 1\}^\kappa$ and any $x \in \{0, 1\}^l$, it holds that

$$\begin{aligned} & \text{Prob} [F, G, H, \dots \leftarrow BC(\kappa, l); ACset \leftarrow \{F, G, H, \dots\}; \\ & \quad O \leftarrow Op[ACset]; r \leftarrow \{0, 1\}^*; (s_1, \dots, s_n) \leftarrow \mathcal{K}(r, k); \\ & \quad y \leftarrow \mathcal{R}(x, s_{i_1}, \dots, s_{i_u}) : Op[ACset](k, x) = y] = 1. \end{aligned}$$

2. *(t, ϵ)-Unpredictability.* For any $A = \{P_{i_1}, \dots, P_{i_u}\} \notin \mathcal{A}$, making t queries to ciphers in $ACset$, it holds that $\text{PRED}_A(Op, \kappa, l, t) \leq \epsilon$, where

$$\begin{aligned} & \text{PRED}_A(Op, \kappa, l, t) = \text{Prob} [F, G, H, \dots \leftarrow BC(\kappa, l); \\ & \quad ACset \leftarrow \{F, G, H, \dots\}; O \leftarrow Op[ACset]; \\ & \quad r \leftarrow \{0, 1\}^*; k \leftarrow \{0, 1\}^\kappa; (s_1, \dots, s_n) \leftarrow \mathcal{K}(r, k); \\ & \quad (x, y) \leftarrow A^{ACset}(s_{i_1}, \dots, s_{i_u}) : Op[ACset](k, x) = y]. \end{aligned}$$

Following the original model of threshold cryptography [7,6], in the above definition we have assumed that the participants honestly follow their protocol. Investigations in threshold cryptography, later than [7,6], have analyzed the case in which the participants can behave in an arbitrarily malicious way. We believe that investigating this case is of interest for our problem as well and leave it open here for further investigations.

3 Definitions: Secret and Sequence Sharing

We review the notion of secret sharing schemes and formally define the new notion of sequence sharing schemes.

Secret sharing schemes. Secret sharing schemes, first introduced in [4,10], are the traditional cryptographic example of how to securely and reliably distribute information. In a secret sharing scheme there are a dealer, n participants, a *distribution algorithm* and a *reconstruction algorithm*. In a first phase, the dealer uses the distribution algorithm to share a secret key k among the n participants via shares $s_1, \dots, s_n$, where share s_i is given to the i -th participant. In a second phase, groups of participants may use their shares and the reconstruction algorithm in order to compute the secret key shared by the dealer in the first phase. The *correctness* requirement of secret sharing schemes says that ‘qualified’ subsets of participants will successfully compute the secret key. The *security* requirement of secret sharing schemes says that ‘unqualified’ subsets of participants receive no information at all about the secret key. The set of qualified subsets of shares is specified by an *access structure*. As usually done in the context of secret sharing schemes, we will only consider *monotone* access structures, namely access structures such that if a subset A is in the structure, so is any other subset who contains A .

Sequence sharing schemes. Informally, a sequence sharing scheme is a method for distributing a sequence of secret keys among a set of participants in such a way that all ‘qualified’ subsets of participants are able to recover the entire sequence, while all ‘unqualified’ subsets of participants receive no information about at least one element of the sequence. Furthermore we require that the shares distributed to participants consist of keys taken from the sequence to be shared. As for the case of secret sharing schemes, the set of qualified parties is specified by a monotone access structure. We now present a formal definition of sequence sharing schemes.

Definition 2. Let n be a positive integer, let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of participants, and let $\mathcal{A}$ be a monotone access structure over set $\mathcal{P}$. A *sequence sharing scheme* for $\mathcal{A}$ is a pair of algorithms: the *distribution algorithm*, called $\mathcal{D}$, and the *reconstruction algorithm*, called $\mathcal{R}$. On input some random string r , and a string $k = (k_1, \dots, k_{m(n)})$ from some key space $K = K_1 \times \dots \times K_{m(n)}$, and for some function m , algorithm $\mathcal{D}$ returns an n -tuple $(s_1, \dots, s_n)$, where $s_i \subseteq \{k_1, \dots, k_{m(n)}\}$, for $i = 1, \dots, n$. On input a tuple $(s_{i_1}, \dots, s_{i_u})$, algorithm $\mathcal{R}$ returns an element from set $K \cup \{\perp\}$.

We say that a pair of algorithms $(\mathcal{D}, \mathcal{R})$ is a *perfectly secure sequence sharing scheme* for access structure $\mathcal{A}$ and for sequences over K if it satisfies the properties of Correctness and Perfect Security, as defined below.

1. *Correctness.* For any $A = \{P_{i_1}, \dots, P_{i_u}\} \in \mathcal{A}$, and any $(k_1, \dots, k_{m(n)}) \in K$, it holds that

$$\text{Prob} [r \leftarrow \{0, 1\}^*; (s_1, \dots, s_n) \leftarrow \mathcal{D}(r, (k_1, \dots, k_{m(n)})) : \\ \mathcal{R}(s_{i_1}, \dots, s_{i_u}) = (k_1, \dots, k_{m(n)})] = 1.$$

2. *Perfect Security.* For any $A = \{P_{i_1}, \dots, P_{i_u}\} \notin \mathcal{A}$, for each distribution $\mathcal{F}$ over K , and all $(s_{i_1}, \dots, s_{i_u}) \subseteq K$, there exists an $i \in \{1, \dots, m(n)\}$ such that for all $v \in K_i$, it holds that $p_0(v) = p_1(v)$, where

$$p_0(v) = \text{Prob} [r \leftarrow \{0, 1\}^*; (k_1, \dots, k_{m(n)}) \leftarrow \mathcal{F}(1^n, m); \\ (sh_1, \dots, sh_n) \leftarrow \mathcal{D}(r, (k_1, \dots, k_{m(n)})) : \\ k_i = v \mid sh_{i_1} = s_{i_1}, \dots, sh_{i_u} = s_{i_u}]; \\ p_1(v) = \text{Prob} [r \leftarrow \{0, 1\}^*; (k_1, \dots, k_{m(n)}) \leftarrow \mathcal{F}(1^n, m) : k_i = v].$$

The value $m(n)$ in Definition 2 is also called the *sequence size* of sequence sharing scheme $(\mathcal{D}, \mathcal{R})$. Note that a subset of parties not in the access structure is not disallowed to obtain, for instance, the value of all but one of the keys in the input sequence that is shared. This argument implies that a sequence sharing scheme for sharing a sequence $k = (k_1, \dots, k_{m(n)})$ may not be a secret sharing scheme for sharing secret key $k' = k_1 \circ \dots \circ k_{m(n)}$. On the other hand, given a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for sharing a sequence $k = (k_1, \dots, k_{m(n)})$ according to access structure $\mathcal{A}$, it is easy to derive a secret sharing scheme $(\mathcal{D}', \mathcal{R}')$ for sharing a secret k' according to access structure $\mathcal{A}$. Specifically, algorithm $\mathcal{D}'$ writes the secret key k' as $k' = k'_1 \oplus \dots \oplus k'_{m(n)}$, for randomly chosen k'_i , and then uses algorithm $\mathcal{D}$ to share the sequence $(k'_1, \dots, k'_{m(n)})$; algorithm $\mathcal{R}'$ uses algorithm $\mathcal{R}$ to recover sequence $(k'_1, \dots, k'_{m(n)})$ and then computes k' by xoring the elements of this sequence.

4 Sequence Sharing vs. Block Cipher Sharing

In this section we show the connection between sequence sharing and block cipher sharing. Specifically, we show that given a sequence sharing scheme and a block cipher (for which, in general, it is not known how to share the computation), it is possible to construct a block cipher sharing scheme. Formally, we obtain the following

Theorem 1. *Let n, l, κ be integers, let F be a block cipher with key length κ and block length l , and, for any function m , define operator $m\text{-Comp} : BC(\kappa, l) \rightarrow BC(m \cdot \kappa, l)$ as*

$$m\text{-Comp}[F]((k_1, \dots, k_{m(n)}), x) = F(k_{m(n)}, F(k_{m(n)-1}, \dots, F(k_1, x) \dots)),$$

for all $x \in \{0, 1\}^l$, and all $k_1, \dots, k_{m(n)} \in \{0, 1\}^\kappa$. Also, let $\mathcal{P}$ be a set of n participants, and $\mathcal{A}$ be an access structure over $\mathcal{P}$, and assume there exists a sequence sharing scheme for $\mathcal{A}$ and for sequences over $(\{0, 1\}^\kappa)^{m(n)}$. If $t \leq 2^\kappa$ there exists a (t, ϵ) -unpredictable block cipher sharing scheme for cipher $m\text{-Comp}[F]$ and access structure $\mathcal{A}$, where $\epsilon = t/2^\kappa$.

In the rest of this section we prove Theorem 1. Informally, our basic approach is to schedule the $m(n)$ independent keys used by the composed cipher $m\text{-Comp}[F]$ through the distribution algorithm of the sequence sharing scheme. In other words, we securely share $m\text{-Comp}[F]$ among the participants by sharing the sequence $k_1, \dots, k_{m(n)}$ of keys for $m\text{-Comp}[F]$ among them through the sequence sharing scheme. Note that all subsets of participants in the access structure $\mathcal{A}$ can compute all keys $k_1, \dots, k_{m(n)}$ and thus compute $m\text{-Comp}[F](x)$ for any $x \in \{0, 1\}^n$. On the other hand, all subsets of participants that do not belong to the access structure $\mathcal{A}$ will get no information about at least one of the keys k_i and therefore will not be able to guess the value of $m\text{-Comp}[F]$ on an input x significantly better than by guessing at random either the output of the value of the key. Now we proceed more formally.

Let us denote by $(\mathcal{D}, \mathcal{R})$ a sequence sharing scheme for access structure $\mathcal{A}$. We now present a block cipher sharing scheme $(\mathcal{K}, \mathcal{E})$ for cipher $m\text{-Comp}[F]$.

The algorithm $\mathcal{K}$. On input parameters $1^n, 1^\kappa$, and a function m , do the following. First, uniformly choose κ -bit keys $k_1, \dots, k_{m(n)}$. Second, run algorithm $\mathcal{D}$ on input $(k_1, \dots, k_{m(n)})$ and let $(s_1, \dots, s_n)$ be its output. Finally, return the tuple $(s_1, \dots, s_n)$ as output.

The protocol $\mathcal{E}$. Each participant P_i is given s_i as additional input and writes s_i as a subset KS_i of $K = \{k_1, \dots, k_{m(n)}\}$. For $i = 1, \dots, m(n)$, let j_i be a fixed index (for instance, the smallest) in $\{1, \dots, n\}$ such that the subset KS_{j_i} of keys given to participant P_{j_i} contains key k_i . Then the participant P_{j_1} , having key k_1 in its subset KS_{j_1} , computes $y_1 = F(k_1, x)$ and sends it to a (not necessarily different) participant P_{j_2} having key k_2 in its subset KS_{j_2} . For $i = 2, \dots, m(n)$, the participant P_{j_i} that has received y_{i-1} , computes $y_i = F(k_i, y_{i-1})$ and sends it to a (not necessarily different) participant $P_{j_{i+1}}$ having key k_{i+1} in its subset $KS_{j_{i+1}}$. Finally, participant $P_{j_{m(n)}}$ can compute $y = y_{m(n)}$. The output of the protocol is set equal to y . If no participant can compute $y_{m(n)}$, the output of the protocol is set equal to $\perp$.

Correctness property of pair $(\mathcal{K}, \mathcal{E})$. The correctness property of $(\mathcal{K}, \mathcal{E})$ follows by using the correctness property of $(\mathcal{D}, \mathcal{R})$. Specifically, the latter property implies that, for each $i = 1, \dots, m(n)$, there exists at least one participant P_{j_i} in $\mathcal{A}$ whose share s_{j_i} contains key k_i . Therefore, any subset $A \in \mathcal{A}$ of participants can recover the sequence $(k_1, \dots, k_{m(n)})$. Now, observe that protocol $\mathcal{E}$ is constructed so to simulate the computation of cipher $m\text{-Comp}[F]$ using interaction between the various parties in A , and therefore, the value y will be correctly computed with probability 1.

Unpredictability property of pair $(\mathcal{K}, \mathcal{E})$. Assume by contradiction that there exists an integer x and a group $A \notin \mathcal{A}$ of participants who can compute $y =$

$m\text{-Comp}[F]((k_1, \dots, k_{m(n)}), x)$ with probability ϵ and making t queries to cipher F . Let $SK_A = (k_{i_1}, \dots, k_{i_u})$ be the sequence of keys among $\{k_1, \dots, k_{m(n)}\}$ that the participants in A are distributed by the dealer and let $d = m(n) - |SK_A|$. We can divide sequence $(k_{m(n)}, \dots, k_1)$ into subsequences $S_0, S_1, \dots, S_d, S_{d+1}$, defined as follows. The (possibly empty) sequence S_{d+1} is the sequence of keys $(k_{m(n)}, k_{m(n)-1}, \dots, k_{e_{d+1}})$, where e_{d+1} is the largest index in $\{1, \dots, m(n)\}$ such that all keys in S_0 are in SK_A and $k_{e_{d+1}-1}$ is not. Similarly, the (possibly empty) sequence S_0 is the sequence of keys $(k_{e_0-1}, \dots, k_2, k_1)$, where e_0 is the smallest index in $\{1, \dots, m(n)\}$ such that all keys in S_0 are in SK_A and k_{e_0} is not. For $j = d, \dots, 1$, the non-empty sequence S_j is constructed as the sequence $(k_{e_{j+1}-1}, \dots, k_{e_j})$, where e_j is the largest index in $\{1, \dots, m(n)\}$ such that key $k_{e_{j+1}-1}$ is not in SK_A , all remaining keys in S_j are in SK_A and key k_{e_j-1} is not in SK_A . We can associate an operator to each of the d sequences $S_1, \dots, S_d$. Specifically, define operator $G_j : BC(\kappa, l) \rightarrow BC(\kappa, l)$ as

$$G_j[F; (k_{e_{j+1}-2}, \dots, k_{e_j})](k, x) = F(k, F(k_{e_{j+1}-2}, \dots, F(k_{e_j}, x) \dots)).$$

Operator G_j can be also seen as operator F composed with a fixed permutation. Therefore, note that the security, in the indistinguishability sense, of the operator G_j is essentially the same as that of the original cipher F (the proof of this fact is simple, but not necessary for the rest of the proof).

We now define operator G as the composition of all operators $G_1, \dots, G_d$. Namely,

$$G[F](k_1, \dots, k_d, x) = G_d(k_d, G_{d-1}(k_{d-1}, \dots, G_1(k_1, x) \dots)),$$

for all $x \in \{0, 1\}^l$ and $k_1, \dots, k_d \in \{0, 1\}^\kappa$. Note that $G[F]$ can essentially be seen as a projection of $m\text{-Comp}[F]$ to a cipher where some of the keys have been fixed to some specific values (namely, those in SK_A). The main result in [1] can be restated as saying that for all adversaries A and all d, κ, l, q, t , it holds that $\text{DIST}_A(d\text{-Comp}, \kappa, l, q, t) \leq (t/2^\kappa)^d$. We note that extending the proof in that paper to the case of G presents no technical difficulty. The assumption that $A \notin \mathcal{A}$ implies that $|SK_A| \leq m(n) - 1$ and therefore $d \geq 1$; we thus obtain the following

Lemma 2. *For any adversary A , all κ, l, t, q such that $t \leq 2^\kappa$, and all $j = 1, \dots, d$, it holds that*

$$\text{DIST}_A(G, \kappa, l, q, t) \leq t/2^\kappa.$$

The (t, ϵ) -unpredictability, for $\epsilon = t/2^\kappa$, of $m\text{-Comp}[F]$ directly follows by combining Lemma 1 with Lemma 2.

5 A Lower Bound Technique for Sequence Sharing

In this section we prove lower bounds on the sequence size of a sequence sharing scheme for several access structures, such as threshold structures, graph-based

access structures, and structures that can be represented as an OR-of-ANDs boolean circuit. In fact, we give a general technique for finding lower bounds for sequence sharing schemes, which can potentially apply to several other access structures. These bounds, as we show in Section 6, turn out to be tight. The presentation is divided as follows. First, in Section 5.1, we present our general lower bound technique. Then, in Section 5.2, we apply the general technique to specific examples of access structures, of particular interest. Because of Theorem 1, any lower bound on the sequence size of sharing schemes gives a lower bound on the number of iterations of the atomic cipher necessary to obtain a block cipher sharing schemes using the composition approach suggested in this paper.

5.1 A General Lower Bound Technique

We show a lower bound on the sequence size for any sequence sharing scheme as a function of the number of subsets that satisfy a certain condition with respect to the access structure. Before stating our result, we need the following definition.

Definition 3. Let S be a set of size n , and let $\mathcal{A}$ be a monotone set of subsets of S . We say that set $IS_{S,\mathcal{A}} \subseteq 2^S$ is a *maximal independent set* associated with S and $\mathcal{A}$, if for any $S_1, S_2 \in IS_{S,\mathcal{A}}$, it holds that $S_1, S_2 \notin \mathcal{A}$, and $S_1 \cup S_2 \in \mathcal{A}$.

The terminology used in defining the above set is taken from its instantiation for the case of graph-based structures, which we will see more in detail later. Now we can state our result as the following

Theorem 2. Let $\mathcal{P}$ be a set of n participants, let $\mathcal{A}$ be an access structure over $\mathcal{P}$. Also, let $(\mathcal{D}, \mathcal{R})$ be a sequence sharing scheme for $\mathcal{A}$, and let $IS_{\mathcal{P},\mathcal{A}}$ be a maximal independent set associated with set $\mathcal{P}$ and access structure $\mathcal{A}$. If $m(n)$ is the sequence size of $(\mathcal{D}, \mathcal{R})$, then it holds that $m(n) \geq |IS_{\mathcal{P},\mathcal{A}}|$.

Proof. We define a function that associates to a set S in $IS_{\mathcal{P},\mathcal{A}}$ the keys that are not received by the parties in subset S after an execution of algorithm $\mathcal{K}$, and prove that any two distinct inputs to such function are mapped to disjoint subsets of keys. The theorem then follows as a direct consequence. Now we proceed more formally.

Let K be the set of keys that are in the sequence that is to be shared according to scheme $(\mathcal{K}, \mathcal{E})$ (i.e., if the sequence contains all distinct keys, then $K = \{k_1, \dots, k_{m(n)}\}$). Let $(s_1, \dots, s_n)$ be the output of $\mathcal{K}$. Note that by definition of sequence sharing scheme it holds that $s_i \subseteq K$, for $i = 1, \dots, n$. Now, define function $g : IS_{\mathcal{P},\mathcal{A}} \rightarrow 2^K$ such that for any $S \in IS_{\mathcal{P},\mathcal{A}}$, where $S = \{P_{i_1}, \dots, P_{i_u}\}$, $g(S)$ is equal to $K \setminus \bigcup_{j=1}^u s_{i_j}$. Now take $S_1, S_2 \in IS_{\mathcal{P},\mathcal{A}}$; and assume by contradiction that $g(S_1) \cap g(S_2) \neq \emptyset$. Recall that by definition of maximal independent set, we know that both S_1, S_2 do not belong to $\mathcal{A}$, and $S_1 \cup S_2 \in \mathcal{A}$. However, by our assumption, we obtain that at least one key is both in the subset of keys $g(S_1)$ and in the subset of keys $g(S_2)$, which implies that there is at least one key that $S_1 \cup S_2$ won't be able to recover out of K . This implies that $g(S_1) \cup g(S_2) \notin \mathcal{A}$, which contradicts the definition of maximal independent set. $\square$

5.2 Lower Bounds for Specific Access Structures

We consider specific access structures that are of particular interest in the literature and show that Theorem 2 can be used to compute a lower bound on the sequence size of any sequence sharing scheme for such structures. Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of participants.

Threshold structures. Let t be a positive integer such that $t \leq n$. The (t, n) -threshold structure is defined as $\mathcal{A}_{t,n} = \{S \mid S \subseteq \mathcal{P}, |S| \geq t\}$. By definition of independent set, we observe that for this structure the set $IS_{\mathcal{P}, \mathcal{A}_{t,n}}$ is the set of all subsets of $\mathcal{P}$ of size $t-1$. Therefore, by Theorem 2 we obtain that the length of the sequence in any sequence sharing scheme for access structure $\mathcal{A}_{t,n}$ is at least the number of subsets of $\mathcal{P}$ of size $t-1$, that is, $\geq \binom{n}{t-1}$.

OR-of-ANDs structures. Let a, b be positive integers such that $a, b \leq n$, let $T_1 = \{P_{i_1}, \dots, P_{i_a}\}$, let $T_2 = \{P_{j_1}, \dots, P_{j_b}\}$ and $T = T_1 \cap T_2$, where $|T| = c$, for some positive integer c . The OR-of-ANDs structure is defined as $\mathcal{A}_{or,a,b,c} = \{S \mid S \subseteq \mathcal{P}, (S = T_1) \vee (S = T_2)\}$. By definition of independent set, we observe that for this structure the set $IS_{\mathcal{P}, \mathcal{A}_{or,a,b,c}}$ can be characterized as the set of subsets that can be written as $X \cup Y$, where either $X = T_1 \setminus \{x\}$, $Y = T_2 \setminus \{y\}$, for all $x \in T_1$ and $y \in T_2$ such that $x, y \notin T$ or $X = T_1 \setminus \{z\}$, $Y = T_2 \setminus \{z\}$, for all $z \in T$. From this characterization, we obtain that $|IS_{\mathcal{P}, \mathcal{A}_{or,a,b,c}}| \geq (a-c)(b-c) + c$. Therefore, by Theorem 2 we obtain that the length of the sequence in any sequence sharing scheme for access structure $\mathcal{A}_{or,a,b,c}$ is greater than or equal to $(a-c)(b-c) + c$.

Graph-based structures. Let $G_n = (V, E)$ be a graph such that $|V| = n$. The G_n -based access structure is defined as $\mathcal{A}_{G_n} = \{S \mid S \subseteq \mathcal{P}, x, y \in S, (x, y) \in E\}$. By definition of independent set, we observe that for this structure the set $IS_{\mathcal{P}, \mathcal{A}_{G_n}}$ can be characterized as the set $MIS(G_n)$ of all maximal and independent subsets of nodes of G_n (namely, those subsets S of nodes such that no two nodes having an edge in G_n are in S , and such that adding one node to S would violate this property). We note that even for simple graphs as a path among n nodes, the size of a maximal and independent set is at least $2^{n/3}$. Therefore, by Theorem 2 we obtain that there exists a graph G_n such that the length of the sequence in any sequence sharing scheme for access structure $\mathcal{A}_{G_n}$ is greater than $2^{n/3}$.

We obtain the following

Theorem 3. *Let $\mathcal{P}$ be a set of n participants, let $\mathcal{A}$ be an access structure over $\mathcal{P}$. For any sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for $\mathcal{A}$, If $m((\mathcal{D}, \mathcal{R}), \mathcal{A})$ is the sequence size of $(\mathcal{D}, \mathcal{R})$, then it holds that*

1. if $\mathcal{A} = \mathcal{A}_{t,n}$ then $m((\mathcal{D}, \mathcal{R}), \mathcal{A}) \geq \binom{n}{t-1}$
2. if $\mathcal{A} = \mathcal{A}_{or,a,b,c}$ then $m((\mathcal{D}, \mathcal{R}), \mathcal{A}) \geq (a-c)(b-c) + c$
3. if $\mathcal{A} = \mathcal{A}_{G_n}$ then $m((\mathcal{D}, \mathcal{R}), \mathcal{A}) \geq |MIS(G_n)| \geq 2^{n/3}$.

6 Upper Bounds for Sequence Sharing Schemes

In this section we show sequence sharing schemes for access structures of special interest in the literature, such as threshold structures, graph-based access structures, and structures that can be represented as an OR-of-ANDs boolean circuit. In all three cases we obtain an upper bound on the sequence length of the specific sequence sharing scheme that exactly matches the lower bound given in Section 5. Because of Theorem 1, any upper bound on the sequence size of sharing schemes gives an upper bound on the number of iterations of the atomic cipher necessary to obtain a block cipher sharing schemes using the composition approach suggested in this paper. The technique used to construct the sequence sharing scheme is similar in all three cases. Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of participants, and recall the definitions of access structures $\mathcal{A}_{t,n}$, $\mathcal{A}_{or,a,b,c}$, $\mathcal{A}_{G_n}$ from previous section.

Threshold structures. Let m be a function defined as $m(t, n) = \binom{n}{t-1}$, enumerate all subsets of $\mathcal{P}$ of size $t - 1$ as $A_1, \dots, A_{m(t,n)}$, and define sequence $k = (k_1, \dots, k_{m(t,n)})$. Then define a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for access structure $\mathcal{A}_{t,n}$ as follows. For $i = 1, \dots, n$, algorithm $\mathcal{D}$ gives to party $P_i \in \mathcal{P}$ all keys k_j such that $P_i \notin A_j$, for $j = 1, \dots, m(t, n)$. Algorithm $\mathcal{R}$ goes as follows. Let $A \subseteq \mathcal{P}$ be such that $|A| \geq t$. Then for any $i \in \{1, \dots, m(t, n)\}$, there exists a $P_j \in A$ such that $P_j \notin A_i$, and therefore P_j has been distributed k_i .

OR-of-ANDs access structures. Let m be a function defined as $m(a, b, c) = (a - c)(b - c) + c$, enumerate all subsets of $\mathcal{P}$ that can be written as $X \cup Y$, where either $X = T_1 \setminus \{x\}$, $Y = T_2 \setminus \{y\}$, for all $x \in T_1$ and $y \in T_2$ such that $x, y \notin T$ or $X = T_1 \setminus \{z\}$, $Y = T_2 \setminus \{z\}$, for all $z \in T$. Let us call these subsets $A_1, \dots, A_{m(a,b,c)}$, and define sequence $k = (k_1, \dots, k_{m(a,b,c)})$. Then define a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for access structure $\mathcal{A}_{or,a,b,c}$ as follows. For $i = 1, \dots, n$, algorithm $\mathcal{D}$ gives to party $P_i \in \mathcal{P}$ all keys k_j such that $P_i \notin A_j$, for $j = 1, \dots, m(a, b, c)$. Algorithm $\mathcal{R}$ goes as follows. Let $A \subseteq \mathcal{P}$ be such that $A \in \mathcal{A}_{or,a,b,c}$. Then for any $i \in \{1, \dots, m(a, b, c)\}$, there exists a $P_j \in A$ such that $P_j \notin A_i$, and therefore P_j has been distributed k_i .

Graph-based access structures. Let m be a function defined as $m(n) = |\text{MIS}(G_n)|$, where recall that $\text{MIS}(G_n)$ is the set of all maximal and independent subsets of nodes of graph G_n . Also, let us enumerate all subsets of $\mathcal{P}$ in $\text{MIS}(G_n)$, as $A_1, \dots, A_{m(n)}$, and define sequence $k = (k_1, \dots, k_{m(n)})$. Then define a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for access structure $\mathcal{A}_{m(n)}$ as follows. For $i = 1, \dots, n$, algorithm $\mathcal{D}$ gives to party $P_i \in \mathcal{P}$ all keys k_j such that $P_i \notin A_j$, for $j = 1, \dots, m(n)$. Algorithm $\mathcal{R}$ goes as follows. Let $A \subseteq \mathcal{P}$ be such that $A \in \mathcal{A}_{m(n)}$. Then for any $i \in \{1, \dots, m(n)\}$, there exists a $P_j \in A$ such that $P_j \notin A_i$, and therefore P_j has been distributed k_i .

We obtain the following

Theorem 4. *Let $\mathcal{P}$ be a set of n participants and let $\mathcal{A}$ be an access structure over $\mathcal{P}$. It holds that:*

1. if $\mathcal{A} = \mathcal{A}_{t,n}$ then there exists a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for $\mathcal{A}$ whose sequence size is equal to $\binom{n}{t-1}$;
2. if $\mathcal{A} = \mathcal{A}_{or,a,b,c}$ then there exists a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for $\mathcal{A}$ whose sequence size is equal to $(a-c)(b-c) + c$;
3. if $\mathcal{A} = \mathcal{A}_{G_n}$ then there exists a sequence sharing scheme $(\mathcal{D}, \mathcal{R})$ for $\mathcal{A}$ whose sequence size is equal to $|\text{MIS}(G_n)|$.

References

1. B. Aiello, M. Bellare, G. Di Crescenzo, and R. Venkatesan, *Security Amplification by Composition: the case of Doubly-Iterated, Ideal Ciphers*, in Proc. of CRYPTO 98.
2. C. Boyd, *Digital Multisignatures*, in Proc. of IMA Conference on Cryptography and Coding, Clarendon Press, 1989.
3. J. C. Benaloh and J. Leichter, *Generalized Secret Sharing and Monotone Functions*, in Proc. of CRYPTO 88.
4. G. R. Blakley, *Safeguarding Cryptographic Keys*, In Proc. of Nat. Computer Conf. AFIPS 1979.
5. R. M. Capocelli, A. De Santis, L. Gargano, and U. Vaccaro, *On the Size of Shares for Secret Sharing Schemes*, in Journal of Cryptology, Vol. **6**, pp. 57–167, 1993.
6. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, *How to Share a Function Securely*, in Proc. of STOC 94.
7. Y. Desmedt and Y. Frankel, *Shared Generation of Authenticators and Signatures*, in Proc. of CRYPTO 91.
8. Y. Desmedt, *Threshold Cryptography*, European Transactions on Telecommunications, 5 (4): 449–457, July 1994.
9. Y. G. Desmedt and Y. Frankel, *Homomorphic Zero-Knowledge Threshold Schemes Over Any Finite Abelian Group*, in SIAM Journal on Discrete Mathematics, 7(4), pp. 667–679, 1994.
10. A. Shamir, *How to Share a Secret*, in Communication of the ACM, vol. 22, n. 11, 1979, pp. 612–613.
11. M. Ito, A. Saito, and T. Nishizeki, *Secret Sharing Scheme Realizing General Access Structure*, in Proc. of IEEE Global Telecommunications Conference, Globecom 87.
12. C. Shannon, *Communication Theory of Secrecy Systems*, in Bell Systems Technical Journal, Vol. **28**, n. 4, pp. 656–715, 1949.
13. D. R. Stinson, *An Explication of Secret Sharing Schemes*, in Design, Codes and Cryptography, Vol. **2**, pp. 357–390, 1992.
14. D. R. Stinson, *Bibliography on Secret Sharing Schemes*, <http://bibd.unl.edu/stinson/ssbib.html>, October 2, 1997.

All Sail, No Anchor, I: Cryptography, Risk, and e-Commerce

Bob Blakley¹ and G.R. Blakley²

¹ Tivoli Systems, Inc., 9020 Capitol of Texas Highway, Building 1, Suite 270, Austin, TX
78759, USA
blakley@tivoli.com

² Department of Mathematics, Milner Hall, Texas A&M University, College Station, TX
77843-3368, USA
blakley@math.tamu.edu

Abstract. This paper examines some ways risk arises in electronic commerce systems, and considers whether and how cryptography can be used to manage these risks. The paper suggests a taxonomy of information artifacts based on the kinds of risks they give rise to. The paper proposes protection regimes for several members of the taxonomy.

1 Business and Risk

More and more, businesses are offering products and services over electronic networks. Doing this generates risks. Electronic products can be stolen, damaged, or destroyed; electronic services can be stolen, corrupted, or disrupted.

In response to a growing awareness of these risks, a variety of security technologies have been developed. Many rely on the application of cryptographic confidentiality and integrity protection to stored or transmitted data.

In this paper we discuss a set of risks to electronic products and services, and investigate how well cryptographic mechanisms manage them.

2 Risk

We'll start with a simple definition of risk, derived from [1]: risk is the probability of a failure multiplied by the consequences of that failure.

$$\text{risk} = \text{probability}(\text{failure}) * \text{consequence}(\text{failure})$$

For the purposes of this paper, we'll consider a failure to be any event which causes a business to lose money through loss of, or damage to, an electronic asset. We won't consider how to measure the probability of a failure. In general an assumption that the probability of a failure is non-zero and changes in a predictable way over time will be enough to make our points.

Our definition of risk will make sense if we can assign a number to the consequences of a failure. In a business context there is a convenient metric: the dollar value of the loss which a failure inflicts on the business. We will adopt this quantity as the definition of the consequence of failure, without going into consideration of how to measure it. Thus risk amounts to the expected dollar loss due to a failure.

3 Value

For failures to have consequences, electronic assets must have value. In today's world lots of them do: electronic cash has direct cash value; digital documents, music, and movies can be sold and therefore have values. Demographic information -- which can be used to generate business opportunities -- has a value. Even a connection to an information system (for example, an Internet Service Provider connection) has a value independent of the value of the information system itself.

4 Intrinsic and Contingent Value

We consider two kinds of value in this paper: intrinsic value and contingent value.

An information artifact has contingent value if a failure "in bit-space" (loss, destruction, modification, or disclosure of the information artifact) causes a loss "in atom-space" (for example, loss or devaluation of a physical asset). Examples of information artifacts with contingent value include:

ownership certificates
coupons (with expiration dates)
contracts

An information artifact has intrinsic value if a failure "in bit-space" is a loss all by itself -- that is, if the information artifact is the asset. Examples of information artifacts with intrinsic value include:

electronic books, music, and movies
e-cash

5 Characteristics of Cryptography

Every cryptographically protected information artifact has three decay properties:

Key decay: Because an attacker can exhaustively search a space of cryptographic keys over time, the "remaining effective strength" of protection applied to an information artifact by the use of encryption with a particular key decreases over time.

Algorithm decay: Because attackers can make mathematical progress on the problems underlying cryptosystems, the "remaining effective strength" of protection applied to an information artifact by the use of encryption using a particular algorithm decreases over time.

Secrecy decay: Because people aren't good at keeping secrets (especially secrets consisting of long strings of random bits) the protection applied to an information artifact by the use of encryption using a particular secret (or "private") key may decrease over time. For example, shares of a secret sharing scheme, or characters of a password, may "leak" over time.

These three properties suggest an obvious law

The law of effectiveness decay:

For every information artifact protected at time i by the application of encryption under a particular key using a particular algorithm, there is a time of failure $f > i$ at which the remaining effective strength of the protection provided by the encryption is zero.

A consequence of the law of effectiveness decay is that at time f , the probability of failure is 1. And at that time the risk is equal to the consequence of failure.

6 Alice Risk

Information artifacts which have contingent value are like Alice through the looking glass - they can be moved from "atom-space" to "bit-space" and back again. When they're in "atom-space" they're not subject to risk in "bit-space", and vice versa.

An example of this is a title deed. Let's say we start with a paper title deed. We can convert the paper form of a title deed into an electronic form. Then we can apply an integrity-protection primitive (like a digital signature, for instance) to it, give it an expiration date, and destroy the paper deed.

While it is in electronic form, it's subject to theft or modification if the encryption can be broken. But now consider what happens when the electronic form of the deed expires. We can use the electronic form of the deed to generate a new paper title deed just before the expiration date. Immediately after this, the electronic deed expires and the new paper form becomes effective. We've taken the deed "out" of "bit-space" and moved it back into "atom-space". If we design our expiration (and record-keeping) mechanisms properly, there's no remaining risk associated with the electronic deed, because it now has zero value (having expired). The risk has been transferred to the new paper deed back in "atom-space".

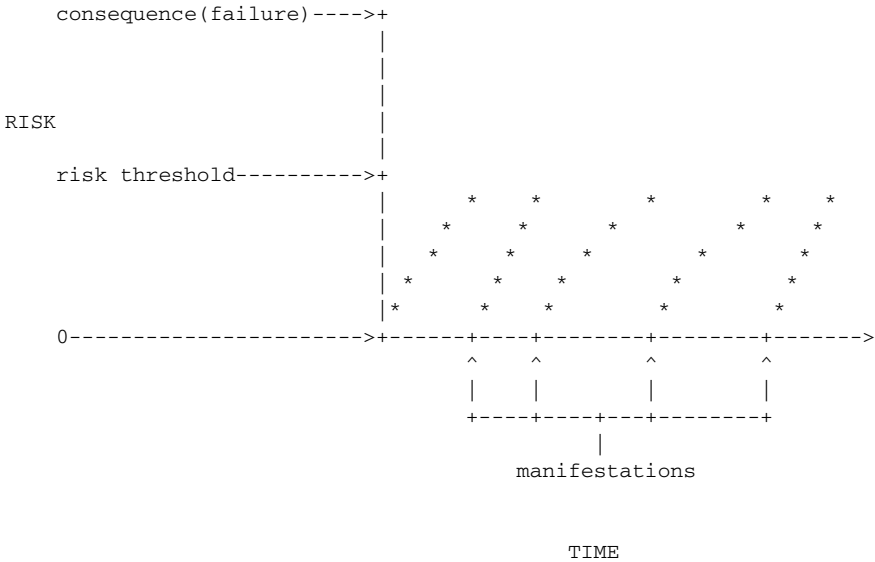
7 Mitigating Alice Risk

The preceding discussion suggests a way to mitigate Alice-type risk: we simply (1) establish a risk threshold, (2) predict the rate of effectiveness decay, and (3) design expiration intervals to insure that the risk never reaches our threshold while the artifact is still in "bit-space". We'll call this technique "periodic manifestation", since the artifact becomes manifest in atom-space periodically, whenever the risk in "bit-space" gets too high.

We should note that in order for periodic manifestation to be effective, it is necessary to design the "bit-space" system so that it has an arrow of time. If an attacker can ever turn back the clock -- so that an expired information artifact appears to be valid -- he can re-introduce risk.

Periodic manifestation may also require bookkeeping to keep track of which version of an Alice-type artifact (the "bit-space" version or the "atom-space" version) is valid at any particular time.

The following diagram shows how risk changes over time for Alice-type information artifacts in a periodic manifestation system:



8 Bob Risk

Information artifacts which have inherent value are not like Alice. They can't move back and forth across the boundary between "bit-space" and "atom-space". Since the authors can't do this either, we'll say that artifacts which have inherent value are like Bob. Once an inherent-value artifact exists in "bit-space", it's stuck there, and it is subject to risk in "bit-space" from that point on.

An example of this is a digital music track. Let's say we start with a studio recording session. We can capture the artist's performance and apply a cryptographic copy-protection primitive to it. Then we can distribute it.

Once we've distributed the protected artifact, an attacker can make a copy of it and keep it indefinitely. The attacker can work as long as he wants in the privacy of his own computer to break the protection which has been applied. And since the artifact has intrinsic value, there's no way we can make it "expire" (unless we're lucky and the musical genre goes out of style in a hurry and the attacker no longer wants to hear the song). The risk cannot be removed from "bit-space" and transferred back to "atom-space".

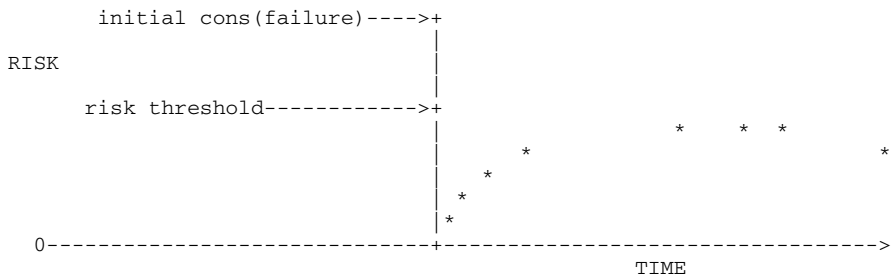
9 Mitigating Bob Risk

The preceding discussion suggests that some Bob-type risks cannot be effectively mitigated. However there is some hope. If a Bob-type information artifact's value decays over time, we may be able to choose sufficiently strong cryptography that by the time effectiveness decays to zero, the value of the artifact has also decayed to zero.

In particular, if a Bob-type information artifact's value depreciates (that is, its value decreases monotonically over time), then we can manage the risk associated with the artifact by (1) choosing a risk threshold, (2) predicting the rate of effectiveness decay of our cryptographic technology, and (3) choosing a strength of initial protection which insures that the risk never exceeds the chosen threshold (though it may approach that threshold asymptotically).

We note that applying cryptographic confidentiality protection to secret data always creates a Bob-type artifact.

The following diagram shows how risk changes over time for depreciating Bob-type information artifacts in a system like the one we've just described.



10 Commodity Risk

We choose to call information artifacts (Alice-type or Bob-type) whose value changes unpredictably "commodities". Our inability to predict the consequences of a failure to protect them makes it difficult to choose either an expiration interval or an initial strength of protection proportionate to the risk they create.

Commodity information artifacts can fall into at least the following four categories:

- A1 Alice-type; value trades within a predictable range during any interval.
- A2 Alice-type; value does not trade within a predictable range for some intervals.
- B1 Bob-type; value trades within a predictable range during any interval.
- B2 Bob-type; value does not trade within a predictable range for some intervals.

This paper argues that the risk associated with commodities of type A1 can be mitigated by assuming that the artifact's value always takes on the maximum predicted value during the current interval, and using periodic manifestation. It might be worthwhile to consider how to manage the risk associated with the other three types of commodities.

References

1. Leveson, Nancy, *Safeware: System Safety and Computers*. Reading, MA: Addison-Wesley, 1995.

Professional Ethics in a Security and Privacy Context - the Perspective of a National Computing Society

Alan Underwood

School of Information Systems,
Queensland University of Technology,
GPO Box 2434, Brisbane 4000, Australia.
a.underwood@qut.edu.au

Abstract. Reference to the words profession and ethics in the media is becoming increasingly common as is the incidence of litigation and exposure by individuals and businesses alike to information technology. Yet, the meaning of the terms profession and ethics remains nebulous for most of us. Perhaps Levin [1] had the right idea in simplifying our understanding when in a speech he used the principle of the camel in illustrating the meaning of a profession: "*few can define it, but none can fail to recognise it.*" This is equally appropriate to ethics.

One of the outcomes arising from the recent Y2K situation was an enhanced awareness by society at large of its dependence on IT systems. Y2K also focussed a spotlight on those working in the IT industry with an expectation that they perform their work in a professional manner. Ethics has always been considered a cornerstone of professional practice. Certainly a policy of *caveat emptor* (let the buyer beware) is inappropriate with respect to the computing profession. It is time to reinforce our professionalism through adherence to a rigorous Code of Ethics specified by a professional IT society so that a policy of *credat emptor* (let the buyer trust) becomes the accepted norm....

1 Definitions

1.1 Ethics

Webster's defines 'ethic' as "the discipline dealing with what is good and bad with moral duty and obligation". Roget's Thesaurus places ethic under morality - to incur a duty, to accept responsibility, to make oneself liable. Reference is also aptly made to a 'conduct of duty' in the context of 'professional status' [2]. Perhaps a more useful description is that ethics are a „framework for human conduct that relates to moral principles and attempt to distinguish right from wrong“ [3].

Rarely will a particular ethical dilemma lend itself to a simple black/white solution. Thus, codes of ethics need to be viewed as a framework to be used in working towards an outcome rather than as a specific solution to a problem.

1.2 Professional

Experience dictates that attempting to define a profession is fraught with great difficulty. Levin [1] a British journalist in a valuable speech given in New Zealand chose instead to rely on the principle of the camel:

"few can define it, but none can fail to recognise it."

The Australian Council of Professions defines a profession as follows:

"A profession is a disciplined group of individuals who adhere to ethical standards and uphold themselves to, and are accepted by the public as possessing special knowledge and skills in a widely recognised body of learning derived from research, education and training at a high level, and who are prepared to exercise this knowledge and these skill in the interest of others.
It is inherent in the definition of a profession that a code of ethics govern the activities of each profession. Such codes require behaviour and practice beyond the personal moral obligations of an individual. They define and demand high standards of behaviour in respect to the services provided to the public and in dealing with professional colleagues.
Further, these codes are enforced by the profession and are acknowledged and accepted by the community."
 (<http://www.austprofessions.com.au>)

The ACP definition above reinforces the role of a professional society such as the ACS in enforcing the adherence of its members to their respective codes of ethics.

Generally it is accepted that a profession can be defined through the identification of a body of common knowledge [4] which embraces the following:

- (i) A standard educational curriculum
- (ii) A requirement to hold an approved tertiary qualification together with relevant experience
- (iii) A Code of Ethics
- (iv) An acceptance of personal liability
- (v) A commitment to ongoing professional development and
- (vi) A licence or certificate to practise

A code of ethics is but one of the necessary requirements for a practising professional in any profession today.

2 Key Ethical Concepts

Mason [5] identified four issues relevant to IT which he summarised with the acronym PAPA which stood for 'privacy', 'accuracy', 'property' and 'accessibility'. Asenjo [6] presents an analysis of fifteen different ethical codes and concludes that "codes of ethics of IS professionals do treat as very important the four ethical issues singled out by Mason."

Another study performed under the auspices of IFIP analysed the codes of ethics of 30 societies of computing professionals [7]. The findings of the IFIP project similarly supported the view that a majority of the codes examined refer to the PAPA issues.

3 The Australian Computer Society

During the period of existence since its formation in 1966 the Australian Computer Society has embraced each of the six points above together with other professional bodies. A recent result of this was the admission of the ACS on 1 January, 2000 into the Australian Council of Professions. Ostensibly the first society of IT professionals to be granted this honour in their own right in the world.

The ACS boasts a current membership of approximately 16,000 and has Branches in each State and Territory of Australia. It is affiliated with international IT bodies such as the International Federation of Information Processing (IFIP) and the South East Asia Regional Computer Confederation (SEARCC).

Professionalism has been a stated priority of the ACS since the author's national presidency in 1990-1991 and has been pursued by subsequent Presidents of the society. The ACS introduced the Practising Computing Professional (PCP) scheme nationally in 1991. The scheme recognises those members who engage in ongoing education activities as Practising Computing Professionals, or PCP's with a certificate issued to individuals recognising this status.

In 1993 the ACS introduced an industry based Certification Program at a masters level consisting of four one semester modules. It is delivered by distance education and can be completed part-time within two years. Examinations are conducted by Deakin University in 95 cities throughout Australia and 33 cities in the Asia Pacific region [acs.org.au].

To complete the Certification Program, participants must pass the two core subjects: IT Trends and Business Legal and Ethical Issues and two subjects from one of the following specialist streams:

- IT Strategy and Management
- Project Management
- Marketing and Selling IT
- e-Business (new in 2000)
- Software Engineering (new in 2000)
- Systems Integration - To be phased out in 2000
- Data Communications - To be phased out in 2000

The ACS supports a number of functional boards similar to the Technical Committee structure of IFIP. The boards most relevant to this topic include the Membership Board, the Community Affairs Board and the Technical Board.

Within the ACS Community Affairs Board the Economic, Legal and Social Implications Committee (ELSIC) addresses ethics & privacy. Recently the committee established a Computer Ethics taskforce while the Committee has for a number of years been proactive in making a number of privacy submissions to the Australian government. For example, a response to the Attorney-General's Department request for comments on 'A privacy scheme for the private sector: Release of Key Provisions' (<http://www.acs.org.au/boards/cab/elsic/privacy-2000-01-agd.html>),

Position on Privacy – (<http://www.acs.org.au/president/1998/past/privpos.htm>), Regulation of the Internet – (<http://www.acs.org.au/boards/cab/regulation.html>) and cryptography as per the IFIP TC11 statement <http://www.acs.org.au/news/caelli.htm>.

All professional members of the ACS are bound by the ACS Code of Ethics (<http://www.acs.org.au/national/pospaper/acs131.htm>) and the Code of Professional Conduct and Professional Practice (www.acs.org.au/national/pospaper/code2.htm). A Disciplinary Committee to act on allegations of improper professional behaviour is maintained as a standing committee by the ACS to take disciplinary action where necessary.

4 The Australian Council of Professions

The initial constitution for the (then) Federal Council of Professions was approved on 12 November 1971. The main objectives of the new Council were seen at that time as being to maintain and advance the standards and status of the professions in the community generally and to uphold and advance the honour and reputation of the professions and the integrity and standing of the members thereof.

The current vision statement published on the Council's web page (<http://www.austprofessions.com.au>) is as follows:

"The ACP, reflecting the needs and commitment of its growing constituency, strives to become the recognised authority on professional ideals, conduct, standards and practice in the service of the community.

It will have an active voice on matters of national development, regulation and education based on well researched information.

It will make good use of modern technologies and will communicate well with government and the community.

The ACP will be efficient, cohesive, dynamic and well-focussed in harnessing the resources of its expanding membership to address the issues of common professional concern." (Adopted, General Meeting, 17 November 1997)

As the national peak body of Australian professional associations the ACP sees its role as advancing and promoting professionalism for the benefit of the community.

5 Professional Standards Bill

In 1999 the New South Wales government introduced the Professional Standards Bill. In essence the bill capped the liability on time and money of a „Professional of Record“ who is listed on an approved register.

The NSW Government has defined a 'Professional of Record' as the professional who authorised the documents or actions related to the matter in contention in an action within the ambit of the legislation. The Australian Council of Professions endorses the incorporation of this definition in the legislative framework, in which an "Approved Register" is defined as:

(a) one in which membership is voluntary and is limited to professionals who

- i. have defined minimum qualifications in both tertiary education and experience such that the holders can operate as professionals independently within their field of competence;
- ii. adhere to a Code of Ethics (the NSW Government's special requirements for risk management and continuing professional development are an inherent part of ethical practice required by the Code); and
- iii. are covered by professional indemnity insurance, either directly or indirectly, to the level required by the legislation; and

(b) one which provides for

- i. eligibility competently assessed using established processes against well defined criteria for tertiary education standards, professional experience and continuing professional development;
- ii. access by the public to a register of current members and ability to lodge complaints;
- iii. disciplinary action for breaches of the Code of Ethics;
- iv. the right of appeal by professionals judged ineligible for entry to the scheme or judged liable for disciplinary action;
- v. access to a mediator or arbitrator, nominated by the President of the Professional Association, to assist in the resolution of disputes; and
- vi. scheme to be run by a "Registration Board" administered by the Professional Association and able to draw on the full range of Professional Association experience and expertise.[<http://www.austprofessions.com.au>]

The Australian Council of Professions further proposes that the following aspects be included in the processes and documentation for a register to be submitted for approval under any legislation:

- i. definition of standards of qualification (both academic and experience) for admission to the register;
- ii. processes for accreditation of educational programs (both Australian and overseas) which satisfy the academic qualification requirement;
- iii. processes for assessment of applicants' conformance to the qualification requirement;
- iv. process for appeal by applicants who have been judged ineligible for entry to the register;
- v. definition of standards to be achieved in continuing professional development;
- vi. processes for assessing conformance of persons on the register to the professional development standards (quality and quantity factors);
- vii. defined Code of Ethics;
- viii. defined requirements for professional indemnity insurance;
- ix. process for issuing of annual practising certificates on payment of the annual fee;
- x. disciplinary processes for breaches of conditions of register;
- xi. process for appeal by persons subject to disciplinary actions;
- xii. processes for mediation and arbitration in disputes; and
- xiii. administrative processes and fee structure for the register.

The Australian Council of Professions expects that, as a result of this practice, registration by the Professions would supersede other registration procedures so that costs on the individual professional would largely be offset. (Adopted, General Meeting, 5 November 1990)

The capping of liability may seem to be some as not necessarily in support of the public interest. The Professional Standards Bill enacted in New South Wales nevertheless illustrates an instantiation of a mechanism for control over a professional society and its members.

The ACS in seeking admission to the ACP was required to demonstrate that the mechanisms were in place to ensure compliance with the requirements of the NSW Professional Standards legislation.

6 Codes of Ethics

From a societal perspective it is comforting to recognise that the major professional computing societies already have codes governing the behaviour of their members. One of the largest professional IT associations, The Council of the Association of Computing Machinery (ACM), adopted a set of guidelines for Professional Conduct in Information Processing as far back as November 1966. Also in the United States the Data Processing Management Association (DPMA) and the Institute of Electrical and Electronics Engineers both have a code of ethics. The British Computer Society (BCS) agreed codes of practice and conduct in 1983 while the Australian Computer Society (ACS) adopted a code of ethics instrument in 1987 which replaced one originally endorsed in 1979.

Many professional IT organisations throughout the world have also introduced similar guidelines. While the intent of these guidelines are usually the same, the contents, wording and titles are often significantly different. In general it would appear that this is related to the character and size of the organisation.

The ACP has defined ethical requirements for their member professions as follows:

Professionals shall:

1. at all times place the responsibility for the welfare, health and safety of the community before their responsibility to the profession, to sectional or private interests, or to other professionals;
2. act so as to uphold and enhance the honour, integrity and dignity of the profession;
3. perform professional practice in only their areas of competence;
4. build their professional reputation on merit and shall not compete unfairly;
5. apply their skills and knowledge in the interest of their employer or client for whom they shall act, in professional matters, as faithful agents or trustees;
6. give evidence, express opinions or make statements in an objective and truthful manner and on the basis of adequate knowledge;
7. continue their professional development throughout their careers and shall actively assist and encourage professionals under their direction to advance their knowledge and experience.

All constituent bodies must have in place disciplinary provision under which alleged breaking of code of ethics by any of their members can be investigated and, if proven, penalised. [Adopted, General Meeting, 5 November 1990]

The recent moves by the Texas Board of Professional Engineers in adopting software engineering as a distinct discipline under which engineering licences could be issued perhaps foreshadows a move towards regulation [8]. It is also interesting to note that a separate Code of Ethics for Software Engineers was developed as a part of this project [<http://www.acm.org/serving/se/code.htm>].

The initiative in Texas is significant since the term engineer in the United States has legal connotations and its use is restricted to those who meet the criteria to be registered as professional engineers.

We still need to recognise however that no regulations currently exist anywhere in the world that require a practising IT professional to be registered, certified, or licensed. A number of specialised certifications exist, for example, Quality, Project Management, Security etc however there is typically no legal requirement binding on anyone practising IT to become certified. In discussing the new software engineering code of ethics Gotterbarn [10] proposes that „Sanctions will occur only when the Code is publicly adopted as a generally accepted standard of practice, and when both society and legislators view the failure to follow the Code as negligence, malpractice, or just poor workmanship“.

7 Security and Privacy

It is probable that the majority of IT professionals have long recognised the importance of including security mechanisms in computer application systems for the protection of personal privacy as well as data. It is also probable that the importance of these issues has been overlooked by computer manufacturers, the user community and society at large. The delivery of computer hardware with security features disabled points to a fundamental ignorance of the importance and application of security in today's electronic world.

The onset of e-business will however highlight the importance of security in the related application systems. Both business and personal users of e-business systems will expect that these systems will guarantee an acceptable level of security and privacy protection. I.T. security professionals will find that their advice is sought and their recommendations implemented. All of a sudden they will be in the spotlight of public attention and will find themselves being held accountable for their actions.

Self regulation by industry with respect to e-business will become extremely important since any inaction will inevitably result in the intrusion of government. This position is ably illustrated by a recent survey of the top 200 most accessed Web sites in Australia. The survey author, Australian National University law faculty masters candidate Ben Macklin, stated „The . . . survey reveals that in an environment without legislation, organisations online have not addressed the privacy and security concerns of the consumer.“ [11]

Other measures such as the recent launch by IBM, Disney and Time Warner of the Online Privacy Alliance, defining a code of practice and developing a privacy symbol

to be carried by Web sites further illustrate the strength of business concerns to privacy. It is likely that the move by these Internet heavyweights was to forestall any potential government intervention. A current proposal in Australia reported by Bushell [11] is for the Government to base its legislation on the Privacy Commissioner's National Principles for the Fair Handling of Personal Information and the draft Data Protection Bill of Victoria. It would involve the adoption of industry codes approved by the Privacy Commissioner. It would also enforce minimum privacy standards in areas where industry codes are not adopted.

Again Macklin's [11] conclusion „that in an environment without legislation, organisations online have not addressed the privacy and security concerns of the consumer" suggests that self-regulation in the industry has not been sufficient and that government legislation is warranted.

Most in industry would accept the view that the greatest returns in doing business electronically relate to business to business transactions. Ensuring a secure environment within which to transfer and process related transaction data is of paramount importance. Similarly there is evidence of privacy concerns related to on-line purchasing being expressed by consumers that is impeding the take-up of e-commerce. The establishment of the necessary operating environment to protect both privacy and security will be heavily influenced by I.T. security professionals. Attention to ethics and adherence to a self-regulatory framework will be essential if the potential of e-Business is to be achieved.

8 Australian Legislation – Privacy and Security

In Australia there are currently no Commonwealth, State or Territory laws that deal with e-Business security nor is there any State or Territory legislation dealing with privacy.

The Australian government enacted the Privacy Act (1988) [12] which applies only to government departments and credit agencies and not to the private sector. The Act embraces ten Information Privacy Principles concerning the protection of privacy and individual liberties espoused by the Organisation for Economic Co-operation and Development.

The failure of the Privacy Act to apply to the Private sector has always been a contentious issue with the development of a Privacy Charter initiated in August 1992 by Justice Michael Kirby, then President of the NSW Court of Appeal; Graham Greenleaf, Associate Professor of Law at the University of NSW; and Simon Davies Director-General of Privacy International. This Charter was launched on 6 December 1994 and sets out 18 general privacy and data protection standards appropriate to the range of issues associated with the right to privacy in the 1990s. In December 1999, the Attorney-General's Department released an information paper on proposed legislation for the protection of privacy in the private sector. At the time of writing this Privacy Amendment (Private Sector) Bill 2000 has been referred to the House Standing Committee on Legal and Constitutional Affairs (<http://www.aph.gov.au/house/committee/laca/Privacybill/inqinf.htm>).

9 Conclusion

Turpeinen [13] in identifying a range of legal and ethical issues related to cryptography and information security concluded that „strong international co-operation is needed to face the global challenges of secure networks and use of cryptography in private and business applications“. This global challenge still has to be met by the I.T. profession.

Yet, the dynamic nature of I.T. and in particular the rapid escalation of electronic trading will result in situations that the current legal system is unable to handle because of an absence of appropriate laws. Again, the national legal systems throughout the world, let alone a global legal body, are ill-equipped to frame laws and legislation in the immediate time-frame necessary. In this vacuum individuals' privacy and the security of business data will rely on the tools, techniques and advice of I.T. security professionals. Given this scenario the protection of privacy and security in installed computing application systems generally will rely largely on the ethical stance of professionals practising in the I.T. industry.

References

1. Levin, Bernard, "Today's Target - The Profession", A speech at the 1986 FIDIC Annual Conference, Auckland, (1986)
2. Sizer, Richard, An IFIP Code of Ethics - Some of the Issues Involved in Education and Society, R. Aiken (Ed.), Information Processing 92, Vol II, Elsevier Science Publishers B.V. (North Holland), IFIP, (1992) 351-354
3. Miesing & Preble referred to in a presentation by a QUT student (1985)
4. Underwood, Alan, The Core Body of Knowledge for Information Technology Professionals, Australian Computer Society (1996)
5. Mason, R.O., Four Ethical Issues of the Information Age in MIS Quarterly, Vol 10, No. 1 (1986) 5-12
6. Asenjo, Porfiro Barrosa, Key Ethical Concepts for the Internet and for Ethical Codes of Computer Professionals in Australian Computer Journal, Vol. 29, No. 1, February (1997) 2-5
7. Berleur, Jacques, Brunnstein, Klaus, (eds.): Ethics of Computing: Codes, Spaces for Discussion and Law, A Handbook prepared by the IFIP Ethics Task Group, London: Chapman & Hall, (1996)
8. Speed, John R., What Do You Mean I Can't Call Myself a Software Engineer? in IEEE Software, November/December (1999) 45-50
9. Gotterbarn, Don, How the New Software engineering Code of Ethics Affects You in IEEE Software, November/December (1999) 58-64
10. Bushell, Sue, Privacy legislation key to e-commerce in Information Age, 1 November (1999)
11. Australian Government Publishing Service, Privacy Act 1988
12. Turpeinen, Marko, Legal and ethical Issues related to Cryptography and Information Security, Tik-110.501 Seminar on Network Security (1995)
http://www.cs.hut.fi/~mtu/netsec/marko_1.html

Author Index

- Abe, Masayuki 163
Anzai, Jun 313
Ashley, Paul 18, 89
Au, Richard 18
- Baek, Joonsang 49
Banks, William D. 378
Bao, Feng 97, 278
Blakley, Bob 471
Blakley, G.R. 266, 471
Borosh, I 266
Boyd, Colin 134, 149, 412
Brickell, Ernie 457
Brown, Lawrie 30
Burmester, Mike 289
- Cant, Tony 206
Chrissikopoulos, Vassilios 289
Clark, Andrew 89, 112
Clark, John A. 242
Cremers, Armin B. 1
- Dawson, Ed 149, 412
Deng, Robert H. 97, 278
Di Crescenzo, Giovanni 457
Doi, Hiroshi 352
- Ferguson, Niels 234
Foo, Ernest 134
Frankel, Yair 457
- Gamage, Chandana G. 74
Gaskell, Gary 89
Ghodosi, Hossein 328
Golić, Jovan Dj. 220
González Nieto, Juanma 149
- Henderson, Marie 206
Hong, Seokhie 41
Hui, Zhu 255
- Jacob, Jeremy L. 242
Jøsang, Audun 191
- Kanda, Masayuki 163
Kim, Kwangjo 49
- King, Brian 443
Klappenecker, A. 266
Kotzanikolaou, Panayiotis 289
Kurnio, Hartono 178
- Lam, Kwok Yan 342
Lee, Byoungcheon 49
Lee, KookHeui 400
Lee, Sangjin 41
Lehmle, Hartmut 1
Leiwo, Jussipekka 74
Lieman, Daniel 378
Lim, Jongin 41
Liu, Chuchang 206
Looi, Mark 18, 89
- Mambo, Masahiro 352
Matsumoto, Tsutomu 313
Matsuzaki, Natsume 313
May, Lauren 112
Mitomi, Shirow 298
Miyaji, Atsuko 298
Miyazaki, Shingo 385
Moon, SangJae 400
Mu, Yi 368
- Nguyen, Khanh Quoc 427
- Okamoto, Eiji 352
Ozols, Maris A. 206
- Pedersen, Ingar Glenn 191
Penna, Lyta 112
Pieprzyk, Josef 328
Polemi, Despina 123
Povey, Dean 191
Prasetijo, Agung 89
- Safavi-Naini, Rei 178
Sakai, Yasuyuki 59
Sakurai, Kouichi 59, 385
Schneier, Bruce 234
Shparlinski, Igor E. 378
Sklavos, Panagiotis 123
Spalka, Adrian 1
Sung, Maenghee 41

- Susilo, Willy 178
- Tan, T.H. Daniel 255
- Traoré, Jacques 427
- Underwood, Alan 477
- Varadharajan, Vijay 368
- Varvitsiotis, Aggelos 123
- Viswanathan, Kapali 149, 412
- Wagner, David 234
- Wang, Huaxiong 178, 342
- Wu, Hongjun 97, 278
- Xiao, Guo-Zhen 342
- Ye, Dingfeng 97, 278
- Yi, Okyeon 41
- Yoo, Heajoung 41
- Zhao, Huanhui 342
- Zheng, Yuliang 74