Damien Sauveron
Konstantinos Markantonakis
Angelos Bilas
Jean-Jacques Quisquater (Eds.)

LNCS 4462

Information Security Theory and Practices

Smart Cards, Mobile and Ubiquitous Computing Systems

First IFIP TC6 / WG 8.8 / WG 11.2 International Workshop, WISTP 2007
Heraklion, Crete, Greece, May 2007
Proceedings

ifip

Springer

Damien Sauveron
Konstantinos Markantonakis Angelos Bilas
Jean-Jacques Quisquater (Eds.)

Information Security Theory and Practices

Smart Cards, Mobile and Ubiquitous Computing Systems

First IFIP TC6 / WG 8.8 / WG 11.2 International Workshop, WISTP 2007
Heraklion, Crete, Greece, May 9-11, 2007
Proceedings

Springer

Volume Editors

Damien Sauveron
XLIM (UMR Université de Limoges/CNRS 6172)
Site Jidé
83 rue d'Isle, 87000 Limoges, France
E-mail: damien.sauveron@xlim.fr

Konstantinos Markantonakis
Royal Holloway, University of London
Information Security Group, Smart Card Centre
Egham, Surrey, TW20 0EX, UK
E-mail: k.markantonakis@rhul.ac.uk

Angelos Bilas
FORTH-ICS
P.O. Box 1385
Heraklion, GR-71110, Greece
E-mail: bilas@ics.forth.gr

Jean-Jacques Quisquater
UCL Crypto Group
Place du Levant, 3
B-1348 Louvain-la-Neuve, Belgium
E-mail: quisquater@dice.ucl.ac.be

Library of Congress Control Number: 2007925691

CR Subject Classification (1998): E.3, K.6.5, C.3, D.4.6, C.2

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-72353-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-72353-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12058549 06/3180 5 4 3 2 1 0

Preface

With the rapid technological development of information technology, computer systems and especially embedded systems are becoming more mobile and ubiquitous. Ensuring the security of these complex and yet resource-constrained systems has emerged as one of the most pressing challenges for researchers. Although there are a number of information security conferences that look at particular aspects of the challenge, we decided to create the Workshop in Information Security Theory and Practices (WISTP) to consider the problem as a whole. In addition the workshop aims to bring together researchers and practitioners in related disciplines and encourage interchange and practical co-operation between academia and industry.

Although this is the first ever WISTP event, the response from researchers was superb with over 68 papers submitted for potential inclusion in the workshop and proceedings. The submissions were reviewed by at least three reviewers, in most cases by four, and for program committee (PC) papers at least five reviewers. This long and rigorous process was only possible thanks to the hard work of the PC members and additional reviewers, listed in the following pages. We would like to express our gratitude to the PC members, who were very supportive from the very beginning of this project. Thanks are also due to the additional expert reviewers who helped the PC to select the final 20 workshop papers for publication in the proceedings. Of course we highly appreciate the efforts of all the authors who submitted papers to WISTP 2007. We hope they will contribute again to a future edition and encourage others to do so.

In this first edition, Prof. Jean-Pierre Hubaux, Prof. Fred Piper, Caspar Bowden and Patrick Waters have honored us with their great experience offering keynote speeches. We want to acknowledge their contribution and amiability.

To host a successful workshop requires not only support from the research community but also practical and financial support from a range of companies and scientific organizations. Therefore, we would like to thank every single one.

Special thanks to Matthieu Finiasz and Thomas Baignères for providing the Web review system iChair that was a great asset for the workshop organization.

March 2007

Damien Sauveron
Konstantinos Markantonakis
Angelos Bilas
Jean-Jacques Quisquater

Organization

WISTP 2007 was organized by FORTH-ICS, ISG-SCC and XLIM.

Workshop Organization Committee

Chair: Damien Sauveron. XLIM, University of Limoges, France

Angelos Bilas. FORTH-ICS and University of Crete, Greece
Konstantinos Markantonakis. ISG-SCC, Royal Holloway University of London, UK

Program Committee

Chairs: Konstantinos Markantonakis. ISG-SCC, Royal Holloway University of London, UK
Jean-Jacques Quisquater. DICE, Catholic University of Louvain, Belgium

François Arnault. XLIM, University of Limoges, France
Angelos Bilas. FORTH-ICS and University of Crete, Greece
Christophe Bidan. SSIR, Supélec, France
Pierre-François Bonnefoi. XLIM, University of Limoges, France
Stefano Campadello. Nokia Research Center, Finland
Serge Chaumette. LaBRI, University Bordeaux 1, France
Tassos Dimitriou. Athens Information Technology, Greece
Pierre Dusart. XLIM, University of Limoges, France
Eduardo Fernández-Medina Patón. Castilla-La Mancha, Spain
Theodoulos Garefalakis. University of Crete, Greece
Dieter Gollmann. TU Hamburg-Harburg, Germany
Stefanos Gritzalis. Info-Sec-Lab, University of the Aegean, Greece
Olivier Heen. Security Laboratory, Thomson R&D, France
Sokratis Katsikas. Info-Sec-Lab, University of the Aegean, Greece
Javier Lopez. Computer Science Department, University of Malaga, Spain
Evangelos P. Markatos. FORTH-ICS and University of Crete, Greece
Fabio Martinelli. Information Security Group, IIT-CNR, Italy
Keith Mayes. ISG-SCC, Royal Holloway University of London, UK
Jan de Meer. Brandenburg Technical University (BTU), Germany
Pierre Paradinas. CEDRIC, CNAM, France
Kenny Paterson. ISG, Royal Holloway University of London, UK
Joachim Posegga. University of Hamburg, Germany
Pierangela Samarati. University of Milan, Italy
Damien Sauveron. XLIM, University of Limoges, France
Paulo Jorge Esteves Veríssimo. LASIGE, University of Lisbon, Portugal

Additional Referees

Asmaa Adnane
Christopher Alm
Kostas Anagnostakis
Spyros Antonatos
Ioannis Askoxylakis
Eve Atallah
Elias Athanasopouylos
Mohamad Badra
Patrick Baier
Lionel Barrère
Stefano Bistarelli
Samia Bouzefrane
Bastian Braun
Arnaud Casteigts
Julien Cordry
Sabrina De Capitani di Vimercati
Estíbaliz Delgado
Alex Dent
Geneiatakis Dimitris
Boris Dragovic
Guillaume Ferré
Louis Goubin
Dimitris Grammenos
Pascal Guitton
Mustapha Hedabou
Guillaume Hiet
Silke Holtmanns
John Iliadis
Sotiris Ioanidis
Martin Johns
Marc Joye
Giorgos Karopoulos
Achraf Karray
Elisavet Konstantinou
Costas Lambrinoudakis
Deok-Gyu Lee
Francois Lesueur
Olivier Ly
Frédéric Majorczyk
Ilaria Matteucci
Benjamin Morin
Marinella Petrocchi
Henrich C. Phls
Michalis Polichronakis
Emanuel Popovici
Nicolas Prigent
Michael Quisquater
Konstantinos Rantos
Evangelos Rekleitis
Rodrigo Roman
Daniel Schreckling
Jacob Schuldt
Jan Seedorf
Jean-Ferdy Susini
Assia Tria
Michael Tunstall
Valérie Viet Triem Tong

Sponsoring Institutions

FORTH-ICS: Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH).
ISG-SCC (Information Security Group - Smart Card Centre) Royal Holloway University of London.
XLIM (UMR Université de Limoges/CNRS 6172).

Main Sponsors

Since the early stages of inception of the workshop, organizers received positive feedback from a number of high-profile organizations. With the development of a strong Program and Organizing committee, this was further capitalized into direct financial support. This enabled the workshop organizers to strengthen significantly their main objective for a high-standard academic workshop. The support helped significantly to keep the workshop registration costs as low as possible and at the same time offer a number of best paper awards. Therefore, we would like to express our gratitude and thank every single organization. We also look forward to working together in future WISTP events.

Table of Contents

A Smart Card Based Distributed Identity Management Infrastructure for Mobile Ad Hoc Networks[⋆]

Eve Atallah[1,2] and Serge Chaumette[1]

[1] LaBRI, Université Bordeaux 1,
351 Cours de la Libération, 33405 Talence Cedex, France
Serge.Chaumette@labri.fr
http://www.labri.fr/perso/chaumett/
[2] XLIM, Université de Limoges,
123, avenue Albert Thomas, 87060 Limoges Cedex, France
Eve.Atallah@labri.fr

Abstract. The network is becoming more and more versatile because of the variety of the computing resources and the communication technologies that have become available. The mobility of the nodes, in these so called Mobile Ad hoc Networks (MANets), furthermore leads to a situation where it is very difficult to establish secure community-based or even peer to peer communication channels. The basic and major problem that has to be solved is that of identity management: how to identify and authenticate an entity that is *a priori* unknown and that tries to dynamically join a community in the network? Even if we solve this problem, how to distribute these certified identities over the network? In this paper, we propose to make a clear distinction between two kinds of organization of a MANet. We consider an identity-based approach and a goal-based approach. In the identity-based approach the nodes of the network have to be precisely identified (i.e. with their real-world identity) and a central administration is therefore required. In the goal-based approach, identities are simply used to distinguish between the nodes that collaborate to a certain goal. We claim that when this second approach is considered, it is possible to support a totally distributed identity management system. Our contribution is the design and the implementation of such a system for these goal-based networks. We assume that the users who want to get involved are provided with PDAs supplied with smart cards and more precisely Java Cards, which are the basic secure bricks on which our approach relies. Of course, our approach supports the uniqueness of identities, but it furthermore enforces permanency, i.e. it prevents changing and repudiation of identity. In this paper, we describe the protocol that we have designed to support our solution and its effective implementation.

Keywords: MANets, identity management, smart cards, auto-administration.

[⋆] This work is partly supported by the ANR/SSIA project Cryscoe and by a PhD grant funded by Region Limousin.

1 Introduction

The need for identity management. Any group (either human or computer based) composed of different and independent entities requires a system to protect its assets and the realization of its founding goal (usually data sharing), by organizing rights and duties management. The success of such a system entirely depends upon the recognition of its components. It is the basis of the management of all security issues [1]. Once recognition is achieved it is for instance possible to create restricted groups with controlled access and to establish secure communication channels.

Specificity of the context. Our framework is that of MANets [2] (Mobile Ad hoc Networks) where, compared to classical networks, there cannot be any central node in charge of the administration and the organization of the network. In this context, the network dynamically evolves, based on the arrival and departure of (the devices of) the final users. Even though it is often claimed that almost nothing can be achieved in such a context, we believe that a lot can be done, provided the nature of such a network is properly taken into account. The basic idea is that there are two ways to consider the organization of a number of entities as a network:

– in **the identity based approach**, a number of nodes work altogether based on the precise knowledge of their respective real-world identities. This can be the case of a group of persons defining a proposal for some project. It might be absolutely necessary to know who is who because there may be confidential information that can only be communicated to specific partners. This is also the case of applications where things are organized around a central system, such as banking systems for instance, where the issuer of an operation must be precisely identified.
– in **the goal based approach**, a number of nodes work altogether so as to achieve a common goal. The question is not to know who is involved, but to make sure that the nodes that are involved contribute to the same precise goal. For instance, in collaborative writing like Wikipedia or in Seti like applications (even though it is in some sens centralized) there is no reason for knowing who is who in the real life. The thing that is important is the goal.[1]

Our approach relies on this subtle difference. We claim that in spontaneous mobile ad hoc networks the first approach is not feasible. Devices need to collaborate independently of any predefined realworld organization. Groups are dynamically built for a specific goal, and the nodes are given identities on the fly, the purpose of which is simply to distinguish them within the group. We can also note that a

[1] This is different from role based approaches [3,4] where users still need to be identified based on their real world identities and where they are given roles by some sort of central administration. The choice of roles comes from the top of the organization, whereas the fact of participating to a goal comes from the bottom.

device might be willing to participate in several activities and thus join different groups, having a different identity in each group.

In all existing solutions, it is mandatory for all the users to be part of a human organization that the network simply reflects. The network entities are dependent of a central administration that must remain available at all time. Our contribution removes these constraints. We provide an auto-administrated architecture that enables the dynamic allocation of identities to the nodes of a MANet; it can then serve as a basis to develop higher level security mechanisms (which are out of the scope of this paper). Our system requires neither centralization of identities nor in-line administration.

2 The Phases of Identity Management, Existing Solutions and MANet Specific Problems

In classical networks (as opposed to MANets), a trusted entity (for instance a dedicated national agency or a network administrator) validates and centralizes the identities of all the users. It delivers ID cards or certificates (X509 [5] for instance). In a MANet, there cannot be any centralization, because of the volatility of the nodes that compose the network and of the network itself. Thus, several specific difficulties appear during the different phases of identity management. In this section, we discuss these difficulties. We present a number of existing approaches that deal with the different phases (validation, certification and distribution) of identity management and explain why they do not solve the problems that we want to address. We deduce some requirements that our solution will need to cope with.

2.1 Nature of the Identities and Their Validation

At the user level, an identity must have a public part which is called the identifier (for instance a login name, a pseudo plus a public key, etc.) and an authenticator that makes the link between the entity and its identifier (for instance a password, a private key, etc.).

Uniqueness. At least the authenticator has to be unique (and it must be kept secret to prevent impersonation). If a central authority generates the keys, the uniqueness is straightforward to achieve. The problem is more difficult to deal with in a MANet, where such a central authority does not exist. It is impossible to compare an identity with all the other identities because there is no global knowledge of the network, i.e. no entity that knows all the identities. A distributed algorithm would not work either, because a node could leave during the verification, or the network could be separated in several disconnected parts. It is thus impossible to validate the uniqueness of an identity in a MANet, unless this uniqueness is guaranteed by the nature of the identity creation process, i.e. by the identity creation algorithm. These considerations lead to the following requirement that we want to support in our solution.

Definition of 1st requirement. An identity must be unique and thus a part of it has to be unique by nature of the creation process. This unique part must be kept secret.

Permanency. Once established, the identity of a device should be definitively linked to this device. When asked to confirm its identity, the device should not be able to deny it. This is the permanency condition that we want to foster.

Definition of 2nd requirement. An identity can neither be modified nor dismissed. The owner of an identity must confirm it when asked for.

A solution to meet this requirement (even though partial since it does not oblige the user to confirm its identity when asked for) is proposed in [6]. Each device is given a pair of asymmetric keys created during its production phase and guaranteed to be unique. Its identifier is the public key of this pair of keys, and its validity is certified by a reliable authority that signs it. The key pair as a whole is kept secret by being stored in a secure module such as a smart card. Identities cannot be changed by the user (the CA would be required) and this is part of the permanency requirement defined above. This solution does not make it possible for a device to have several identities, it does not support non repudiation, and it depends on an administration infrastructure. It thus does not meet our goals.

2.2 Certification and Distribution

Once the identities have been generated and validated, the next step consists in distributing them along with the the proof that they are valid, this assembled information being usually referred to as a certified version of the identity. Several methods exist to achieve these operations. In the rest of this section, we describe two of the major approaches that could at first glance be considered as potential solutions to solve the problem in a MANet.

Auto-organization: to mitigate the problem of the availability of nodes for the distribution of certificates, solutions such as those described in [1,7,8] rely on the use of a trusted node chosen using a method to establish confidence [9]. Establishing confidence requires to observe a number of nodes over a period of time and thus to recognize them. This kind of approach can therefore only be used once identities have been validated, which is precisely what we are trying to do. It thus cannot help in our context.

Signature by a Certification Authority (CA): in classical networks, the authority is centralized on one or several nodes that share the same pair of primary keys. For large networks this approach is extended based on a hierarchical organization. As MANets cannot rely on the continuing existence of any specific node, solutions to distribute this otherwise central authority were developed that use the *threshold secret sharing* principle. The key of the CA is shared by a set of n nodes and $k < n$ partial signatures are required to reconstruct a complete signed certificate. These methods allow to admit a new node in the network as the result of the collective decision of at least k nodes. There are several partial signatures protocols that use RSA [10,11,12,13] or DSA [14,15] keys. The distribution of the authority can be partial [16,17] or total [18,19], in which case every node is supplied with a partial key.

The advantages of these solutions are as follows. First, to compromise the system several nodes must be attacked (get the CA secret key, DOS, etc.) instead of only one. Second, it increases the global availability of the certification authority (by choosing n large enough compared to k). If the distribution is partial, some devices are in charge of a more important task and thus have a specific non symmetric role, what we do not want. We furthermore have the problem of locating these nodes that share the authority. If the distribution is total there is equality between the nodes and no more localization question. In any case, an initialization step is necessary that requires the presence of an administrator and a certain number of nodes for the initial distribution of keys. The administration phase is then strongly dependent of the network specific usage and must be initiated before the network is constituted, what clearly removes the spontaneity, which is a feature that we want to support. Therefore our third requirement.

Definition of 3rd requirement. The validation, certification and distribution phases cannot rely on a central administration that would reflect an ***a priori*** restricted human (i.e. real-world) organization.

3 Our Solution

Based on the requirements defined above we can establish a number of features to achieve at the implementation level.

1st requirement. An identity must be unique and thus a part of it has to be unique by nature of the creation process. This unique part must be kept secret.

To implement this constraint, we need a process to generate unique identities and the possibility to store them in a secure, read only area.

2nd requirement. An identity can neither be modified nor dismissed. The owner of an identity must confirm it when asked for.

This requires the capability to store data in a non erasable, read only area. The confirmation of an identity must be out of the control of its owner, so that he cannot deny it or wrongly confirm an identity that he does not own.

3rd requirement. The validation, certification and distribution phases cannot rely on a central administration that would reflect an ***a priori*** restricted human (i.e. real-world) organization.

As a consequence of this 3rd requirement we have to make virtual the notion of CA.

Presentation of Java Cards. Our solution satisfies these requirements by using smart cards and more precisely Java Cards[2] which provide some specific features, among which:

– their ROM memory makes it possible to install applications (in factory) that can thereafter neither be modified nor erased. Note that the ROM of the cards cannot store application data.
– they allow to store persistent data.
– the information they store can be protected by a firewall that sits between applications.

We strongly rely on the fact that smart cards are secured devices and we do not consider physical attacks like fault injection.

We have specifically chosen Java Cards rather than any other brand of card because they are easy to program and we have been using this technology for quite a long time in our team.

In the rest of this section we give an overview of our solution and how it is used. The lower level protocol is described in section 4.

Implementation of our solution. Each card is prepared as follows:

1) An applet (Java Card application) is installed in factory on each card. It provides the methods required to define identities and to ensure their definitive registration. The use of data stored inside a card is completely controlled by this card and is thus limited by nature to the operations that we have defined. It is then possible to register an identity permanently without any risk that it is modified or erased.
2) Each card stores (installed in factory) a global public key, a specific and unique asymmetric key pair and the public key of this pair signed by the related global private key. These data can thereafter never be accessed from outside the card. Even if smart cards are considered really safe, we still want to ensure that if a card were after all compromised, security of all the identity management architecture would not fall. Therefore we do not store a secret symmetric key or a private asymmetric key[3].

This secure platform is then used as follows:

1) To communicate its identity to another card when asked for, a card encrypts it with its own private key and supplements it with its signed (by the global private key) public key. This is enough to prove that this identity has been

[2] Java and all Java-based marks are trademarks or registered trademarks of Sun microsystems, Inc. in the United States and other countries. The authors are independent of Sun microsystems, Inc. All other marks are the property of their respective owners.

[3] If we had ignored this risk, we could have chosen to include the global private key in each card and then to generate the card key pair and to sign its public key inside the card.

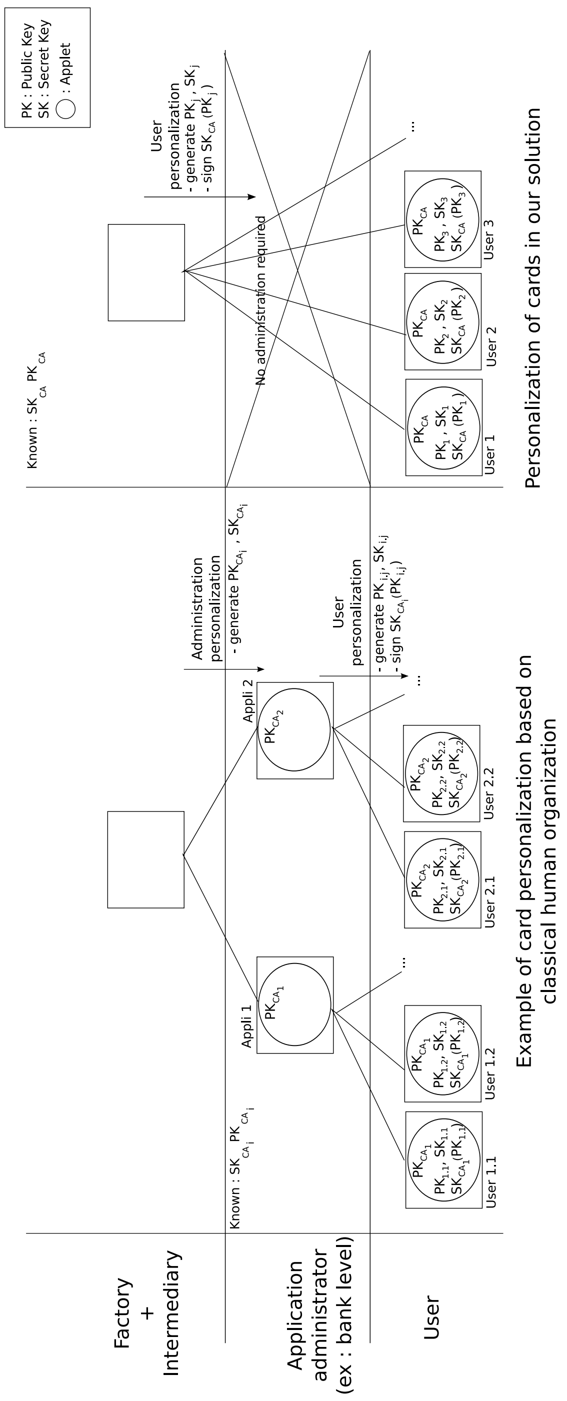

Factory + Intermediary
Application administrator (ex : bank level)
User
Known : SK_{CA_i} PK_{CA_i}
Appli 1
Appli 2
PK_{CA_1}
PK_{CA_2}
Administration personalization - generate PK_{CA_i}, SK_{CA_i}
User personalization - generate $PK_{i,j}$, $SK_{i,j}$ - sign $SK_{CA_i}(PK_{i,j})$
User 1.1
User 1.2
User 2.1
User 2.2
PK_{CA_1} $PK_{1.1}$, $SK_{1.1}$ $SK_{CA_1}(PK_{1.1})$
PK_{CA_1} $PK_{1.2}$, $SK_{1.2}$ $SK_{CA_1}(PK_{1.2})$
PK_{CA_2} $PK_{2.1}$, $SK_{2.1}$ $SK_{CA_2}(PK_{2.1})$
PK_{CA_2} $PK_{2.2}$, $SK_{2.2}$ $SK_{CA_2}(PK_{2.2})$
...
Example of card personalization based on classical human organization
Known : SK_{CA} PK_{CA}
No administration required
User personalization - generate PK_j, SK_j - sign $SK_{CA}(PK_j)$
User 1
User 2
User 3
PK_{CA} PK_1, SK_1 $SK_{CA}(PK_1)$
PK_{CA} PK_2, SK_2 $SK_{CA}(PK_2)$
PK_{CA} PK_3, SK_3 $SK_{CA}(PK_3)$
Personalization of cards in our solution
PK : Public Key
SK : Secret Key
: Applet

Fig. 1. Compared personalization stages

originally provided from inside a card and that it obeys the rules that we have defined (uniqueness, permanence) [4].

2) This could then be used to exchange a session key to achieve secure communication. This would lead to good forward secrecy and fast throughput between smartcards[5].

Our solution vs other smart card infrastructures. Even though smart cards are already widely used for identification and authentication between a user and a central system, there are many differences with the context of our approach. The global key pair, the public key of which is installed on all the cards, does not depend on a specific application that the card could be used for, and thus the personalization of the card for a specific application does not take place at the same level as in classical solutions. This is a consequence of our discussion on identity vs. goal based approaches in section 1 page 2, and is described figure 1.

Advantages of our solution (See figure 2). We have established an auto-administration system that implements the identity management requirements that we have defined. It supports a high level of security, leaves the user total freedom, and requires no preparation once the cards are out of the factory. No user or administrator of the network has to care about key management. The final user only needs to provide an identifier. Furthermore, even once supplied with an identity, a card is not dedicated to one single application as it is the case for instance with banking cards that are specific to one single bank. It can be supplied with new identities and join other goal-based networks on the fly.

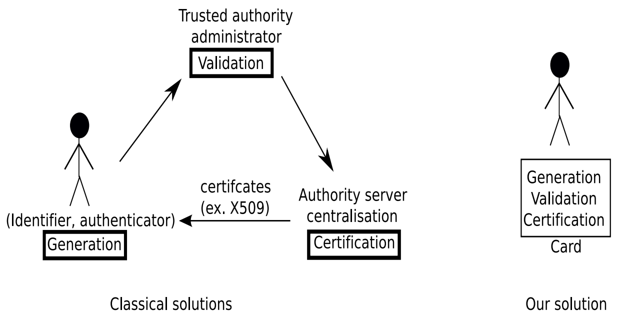

Trusted authority administrator
Validation
(Identifier, authenticator)
Generation
certifcates (ex. X509)
Authority server centralisation
Certification
Generation Validation Certification
Card
Classical solutions
Our solution

Fig. 2. A comparison of solutions for identity management in MANets

[4] It should be noted that if a card becomes compromised and the global public key is discovered, the attacker can get to know the identities of the entities that participate in the network. This has no impact over the security or the proprieties of our infrastructure. Furthermore, the fact that cards are considered today as the most trusted available devices makes it possible to ignore this hypothetical risk.

[5] This precision was suggested by one of the reviewers of this paper.

4 Our Protocol

In our protocol, all communications between cards obey a number of rules:

1. All messages are encrypted by the private key of the sending card.
2. So that a message can be deciphered, it contains the public key of the sending card signed by the private global key (that signed key was stored in every card in factory).
3. A message also contains the following information:
   – the nature of the request (for instance «ACert» for a certificate request).
   – a nounce used to avoid replay. This for instance prevents repeated storage operations that would saturate the memory of cards.

Additional data can then be added according to the type of the message.

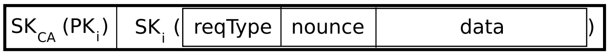


Fig. 3. Structure of a message

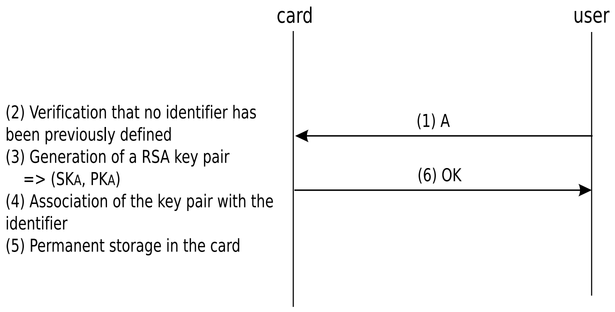

card
user
(1) A
(6) OK
(2) Verification that no identifier has been previously defined
(3) Generation of a RSA key pair => (SKA, PKA)
(4) Association of the key pair with the identifier
(5) Permanent storage in the card

Fig. 4. Creation of an identity

Creation of an identity (see Fig. 4). The user first provides the card with an identifier (1). The card then verifies if an identifier has already been defined (2) and if not, generates a RSA key pair (3), associates it with the received identifier (4) and definitively loads the association in the card (5).

This complies with our 1st requirement, since thanks to the RSA key that it contains, the identity is unique by creation. The identity is protected by the card and its private part is kept secret inside the card. Permanence which is our 2nd requirement is also supported because we provide no method to modify or erase an identity[6]. This approach also complies with the 3rd requirement,

[6] Note that swapping a smart card for a new one to get a new identity would have no consequence over the global infrastructure. This precision has been suggseted by a remark of one of the reviewers of this paper.

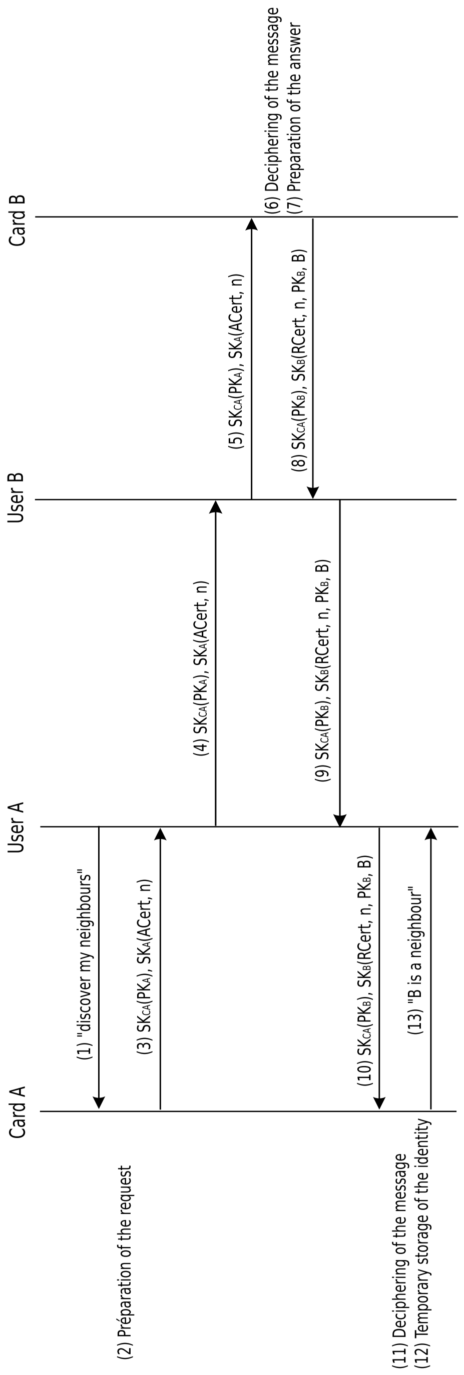

Card A
User A
User B
Card B
(1) "discover my neighbours"
(2) Préparation of the request
(3) $SK_{CA}(PK_A)$, $SK_A(ACert, n)$
(4) $SK_{CA}(PK_A)$, $SK_A(ACert, n)$
(5) $SK_{CA}(PK_A)$, $SK_A(ACert, n)$
(6) Deciphering of the message
(7) Preparation of the answer
(8) $SK_{CA}(PK_B)$, $SK_B(RCert, n, PK_B, B)$
(9) $SK_{CA}(PK_B)$, $SK_B(RCert, n, PK_B, B)$
(10) $SK_{CA}(PK_B)$, $SK_B(RCert, n, PK_B, B)$
(11) Deciphering of the message
(12) Temporary storage of the identity
(13) "B is a neighbour"

Fig. 5. Certification and distribution of identities

since the validation of identities does not require any *a priori* restricted central administration.

Certification and distribution (see Fig.5). First, the user sends a request to the card asking it to discover his neighbours (1). The card prepares the request (2) that only contains the request type and the nounce. It is sent (ciphered) to the neighbourhood (3)(4)(5). A card that receives this request deciphers it (6) and prepares an answer by adding its public identity (public key + identifier) to the message (7). The response is sent (ciphered) (8)(9) and the initial card receives all these incoming messages (10). It deciphers them (11) and temporarily stores all the received identities (12). The identifiers are then propagated to the user level (13).

In terms of robustness and security, the situation is as follows: if the communication breaks at stage (4), there are no consequences (the remote card will simply remain undiscovered). If communication breaks at stage (9), there is no real consequence either since nobody has stored anything yet and it thus cannot be an attack to saturate any of the cards. Once again, the remote card will simply not be discovered.

Once all these steps have been achieved, the deployed validated and certified identities can be used to enable secure communications.

5 Conclusion

In this paper we have presented an identity management system that we have designed for entities willing to collaborate in a goal based approach over a MANet. A prototype has been implemented on a number of Dell Axim PDAs and a draft video demo can be seen on the web at [20].

This identity management architecture sets a basis to establish higher level security features. One of its main characteristics and advantages is that it does not impose any constraint on the natural spontaneity of such dynamic networks. Thanks to the use of Java Cards, the creation and storage of (certified) identities make it possible to support the basic security requirements that we have defined (uniqueness and permanency), without any central administration or server. The fact that all the administration takes place inside the card makes the nodes of the network completely independent of any preexisting real-world group or organization. Every user (or node) thus has the possibility to create a group without any human intervention, wherever he wants, whenever he wants. This is one of the outcomes of the clear distinction that we have made between identity based networks and goal based networks.

Future work directions concern the way the goal of a group is defined and the way a node willing to join a group is allowed to enter it. Basically, the goal will be described by means of a *charter* [21] that contains a number of questions the node will have to answer. This work on charters is part of the MADNESS project carried out at XLIM, University of Limoges. Once this will be achieved, we will be able to conduct an evaluation of the global system.

References

1. Hubaux, J., Buttyan, L., Capkun, S.: Self-organized public-key management for mobile ad hoc networks. In: Proceedings of the ACM International Workshop on Wireless Security. Volume 2., IEEE Transactions on Mobile Computing (january 2003) 52–64
2. Chlamtac, I., Conti, M., Liu, J.: Mobile ad hoc networking: Imperatives and challenges. Elsevier Ad Hoc Networks Journal **1** (july 2003) 13–64
3. Ferraiolo, D., Cugini, J., Kuhn, D.: Role based access control: Features and motivations. In: Proceedings of the 11th Annual Conference on Computer Security Applications, IEEE Computer Society Press, Los Alamitos, CA (1995) 241–248
4. Ferraiolo, D., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed NIST standard for role-based access control. Information and System Security **4** (august 2001) 224–272
5. IETF: ITU-t recommendation X.509 (revised) - information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks (2000) ISO/IEC 9594-8.
6. Kargl, F., Schlott, S., Weber, M.: Identification in ad hoc networks. In: Proceedings of the 39th Annual Hawaiian International Conference on System Sciences. Volume 9., IEEE Computer Society, Washington, DC (January 2006)
7. Capkun, S., Hubaux, J., Buttyan, L.: Mobility helps security in ad hoc networks. In: Proceeding of the 4th ACM international Symposium on Mobile Ad Hoc Networking and Computing MobiHoc'03, Annapolis, Maryland, USA, ACM Press, New York, NY (June 2003) 46–56
8. Garfinkel, S.: PGP : Pretty Good Privacy. O'Reilly & Associates, Sebastopol, California (1995)
9. Marias, G., Papapanagiotou, K., Tsetsos, V., Sekkas, O., Georgiadis, P.: Integrating a trust framework with a distributed certificate validation scheme for MANETs. EURASIP Journal on Wireless Communications and Networking (2006)
10. Y. Frankel, P. Gemmel, P.M., Yung, M.: Proactive RSA. In: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology (Crypto'97). Volume 1294., Lecture Notes In Computer Science (August 1997) 440–454
11. Y. Frankel, P.M., Yung, M.: Adaptive security for the additive-sharing based proactive RSA. In: Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography. Volume 1992., Lecture Notes In Computer Science (February 2001) 240–263
12. Rabin, T.: A simplified approach to threshold and proactive RSA. In: Proceedings of the 18th Annual international Cryptology Conference on Advances in Cryptology (Crypto'98). Volume 1462., Lecture Notes In Computer Science (august 1998) 89–104
13. Shoup, V.: Practical threshold signatures. In: Proceedings of EUROCRYPT'00. Volume 1807., Lecture Notes in Computer Science (2000) 207–220
14. M. Narasimba, G.T., Yi, J.: On the utility of distributed cryptography in P2P and MANets : the case of membership control. In: Proceedings of the 11th IEEE International Conference on Network Protocol (ICNP), IEEE Computer Society, Washington, DC (november 2003) 336–345
15. N. Saxena, G.T., Yi, J.: Admission control in peer-to-peer : Design and performance evaluation. In: Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Network (SASN'03), Fairfax, Virginia, ACM Press, New York, NY (october 2003) 104–113

16. L. Zhou, Z.H.: Securing ad hoc networks. IEEE Network **13** (november/december 1999) 24–30
17. Yi, S., Kravets, R.: MOCA : Mobile certificate authority for wireless ad hoc networks. In: Proceedings of the 2nd Annual PKI Research Workshop (PKI'03). (2003)
18. Luo, H., Lu, S.: Ubiquitous and robust authentication services for ad hoc wireless networks. Technical report, Dept. of Computer Science, UCLA (October 2000)
19. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing robust and ubiquitous security support for mobile ad-hoc networks. In: Proceedings of the 9th International Conference on Network Protocols (ICNP'01), IEEE Computer Society, Washington, DC (2001) 251–260
20. Atallah, E., Chaumette, S.: http://www.labri.fr/perso/chaumett/pda_0001.wmv.
21. Atallah, E., Bonnefoi, P.F., Burgod, C., Sauveron, D.: Mobile ad hoc network with embedded secure system. In: Proceedings of the seventh edition of e-Smart conference, Nice, France (September 2006)

A New Resilient Key Management Protocol for Wireless Sensor Networks

Chakib Bekara and Maryline Laurent-Maknavicius

Institut National des Télécommunications d'Evry
Département Logiciel-Réseaux
9 rue Charles Fourier, 91000 Evry Cedex, France

Abstract. Wireless Sensor Networks (WSN) security is an important issue which has been investigated by researchers for few years. The most fundamental security problem in WSN is key management that covers the establishment, distribution, renewing and revocation of cryptographic keys. Several key management protocols were proposed in the literature. Unfortunately, most of them are not resilient to nodes capture. This means that an attacker compromising a node can reuse the node's key materials to populate any part of the network with cloned nodes and new injected nodes. In this article, we present a simple polynomial-based key management protocol using a group-based deployment model without any necessary predictable deployment location of nodes. That solution achieves high resilience to nodes compromising compared with other protocols.

Keywords: WSN Security, Key Management, Nodes Compromising, Intrusion Detection.

1 Introduction

Wireless sensor networks (WSN) are infrastructure-less and self-organizing networks, which can be deployed anywhere, and work without any assistance [1]. These characteristics motivated their deployment, but introduced critical security issues like network control access, authentication, confidentiality and nodes compromising. WSN are very sensitive to those issues since sensors are known to be tamper-vulnerable devices [2], and deployment of them is mostly done in open area that should be assimilated as hostile area for security consideration.

Current WSN security solutions rely on secret keys but today an efficient key management protocol is still needed to generate, distribute, renew and revoke cryptographic keys. In the last few years, several key management protocols for WSN have been defined, but they do not satisfy the protocol efficiency requirements as follows:

1. Low storage, computation, and transmission overheads.
2. Resistance to nodes compromising, so keys established between non compromised nodes remain confidential even in case of nodes compromising.

3. No on-line key management server, that would be the point of failure in case of DoS attacks.
4. Resilience to nodes compromising that prevents attackers to populate the network with clones of compromised nodes or injected false nodes, by reusing their key materials.

Most of the key management protocols [2] [3] [4] [5], satisfy one or more of the three first requirements. Unfortunately the last requirement is rarely or not enough investigated. Most of the key management protocols are poorly resilient to nodes compromising, and the few ones achieving an acceptable level of resilience, either rely on strong assumptions (i.e. nodes are tamper-resistant) [6] [4], introduce heavy overheads (the use of public key cryptography) [6], or require prior knowledge on nodes deployment [4].

In this article, we propose a simple key management protocol for static WSN, based on the well-known polynomial-based key generation protocol of [7] for pair-wise keys establishment, and our proposed group-based deployment model to ensure resilience to nodes compromising. Our protocol requires no prior knowledge on the locations of deployed nodes. It relies on realistic assumptions, and introduces no significant overhead.

The remainder of the paper is organized as follows. In section 2, we introduce the basic version of polynomial-based key generation protocol. In section 3, we present our assumptions, network model, notations and we define our group-based deployment model. In sections 4 and 5, we describe our proposed protocol, and in section 7 we give its detailed security analysis. In section 8, we briefly review related works on key management in WSN, and we compare the resiliency of our protocol to the resiliency of the related works in section 9 , and we conclude our work in section 10.

2 Polynomial-Based Key Generation Protocol

Blundo et al. [7] present a new pair-wise key establishment protocol, based on a symmetric bivariate polynomial. A trusted key server in the network generates a symmetric bivariate polynomial, as follows:

$$f(x,\ y) = \sum_{i,j=0,\ldots,t} a_{ij} x^i y^j \pmod{Q} \tag{1}$$

where Q is a sufficiently great prime number, $1 \leq a_{ij} \leq Q-1$, and t is the degree of the polynomial and a security parameter. Initially, the key server configures each node u with its unique secret polynomial share:

$$f_u(y) = f(Id_u,\ y) = \sum_{i,j=0,\ldots,t} a_{ij} (Id_u)^i y^j \pmod{Q} \tag{2}$$

where Id_u is the unique identifier of node u in the network. Two nodes of the network u and v, can easily establish a unique shared secret key by computing:

$$K_{uv} = f(Id_u,\ Id_v) = f(Id_v,\ Id_u) = K_{vu} \tag{3}$$

As long as the number of compromised nodes in the network is less than $t + 1$ nodes, the system is secure. The greater the t-value is, the more secure (resistant to nodes compromising) the system is.

3 Assumptions, Notations and Network Model

3.1 Assumptions

First, we suppose that the Base Station (BS) is a *trusted* and a *powerful* entity in the network, that cannot be compromised.

Second, we suppose that sensors are static, so once they are deployed they do not leave their locations. In many scenarios (i.e. perimeter monitoring), WSN are considered as static, either because sensors are fixed or because sensors are not asked to be mobile for achieving their tasks.

Third, we suppose that sensors are deployed progressively in successive generations (groups). This assumption is adopted in most group-based deployment key management protocols like [3] and [8]. However, unlike the other group-based key management protocols, we do not suppose that nodes of the same generation are deployed in the same neighborhood. In our protocol, nodes of the same generation might be deployed anywhere in the network. Therefore, our protocol is not based on any prior knowledge on deployment location of nodes, but if such information was available, our protocol will achieve better resilience.

Fourth, we suppose that an attacker needs a minimum time T_{comp} in order to compromise a node after it is deployed. T_{comp} is greater than the time T_{est}, which is the maximum time needed by a newly deployed node to establish pair-wise keys with its one-hop neighbors. This assumption is present in several works like [9] and [5], and is likely to be true, because an attacker must first have a physical access to a sensor, and then use some programming tools in order to retrieve sensor's key materials. However, in [9] and [5], deployed nodes are initially loaded with some common secrets that nodes use to establish different keys (pair-wise keys, cluster keys) with their neighbors. In addition, [9] and [5] require that each newly deployed node erases these common secrets after a time T_{est} from its deployment, to prevent that an attacker can get them if it is compromised. In our protocol, no such assumption exists, because each node only needs its unique secret polynomial share for pair-wise key establishment.

Fifth, we suppose that sensors are synchronized with the BS. This could be done through an authenticated beacon periodically broadcasted by the BS, to keep sensor's clocks synchronized with the BS's one. Authentication can be guaranteed using the $\mu Tesla$ protocol [10].

Finally, we suppose that an attacker can only get a partial control over the network. In case of full control on all deployed nodes, security solutions will be inoperative to stop the attacker.

3.2 Notations

Table 1 summarizes the notations used in this paper.

Table 1. The used notations

Notation	Significance
u, v	Two nodes of the WSN
Id_u	4 bytes unique identifier of node u in the network.
N_u	An increasing nonce value generated by node u
f_u	The secret polynomial share of node u
$K_{uv} = K_{vu}$	The secret pair-wise key established between u and v
$MAC_K(M)$	The message authentication code of M using the secret key K
H	A one way hash function
$\|x\|$	The length on bytes of x
$a\|\|b$	a concatenated to b

3.3 Network Model and Security Considerations

The BS deploys nodes in multiple generations numbered successively from 1 to n, where n is the maximum number of deployed generations. If we suppose that $n < 2^{16} - 1$, each generation's number is identified by exactly 2 bytes. The order of deployment must be respected $G_1, \ldots, G_i, G_{i+1}, \ldots, G_n$, where G_i is the i^{th} deployed generation. Each node belongs to a unique generation.

Because nodes are not mobile in our network, it is logical that *only* nodes of the newly deployed generation ask for key establishment with their neighbors, which may belong either to the same generation, or to former deployed generations. Nodes of former generations *can not* request for key establishment, and even if they do request, their requests must be rejected. Based on this assumption, we can state that any key establishment request originates from:

- either a node from the newly deployed generation,
- or a node deployed by an attacker, which is either a node having a false Id, or a cloned node having the Id of a compromised node.

For security reasons, we suppose that any newly deployed node u sets a timer to the value T_{est} straight after deployment. Once the timer expires, node u rejects any key establishment request originating from a node of the same newly deployed generation.

4 Our Proposed Protocol

We propose a resilient key management protocol, based on the use of a symmetric polynomial for secure key establishment, and based on our defined group-based deployment model for achieving resilience to nodes compromising. Our protocol involves three phases: *initialization*, *pair-wise key establishment* and *key-path establishment*.

4.1 Initialization Phase

Initially, the BS generates a random symmetric bivariate polynomial $f(x, y)$ (see (1)). The BS then selects a group of nodes to form the next deployed sensors generation. The BS loads each node u with a unique secret polynomial share (see (2)) as follows: suppose $u \in G_i$, then the BS loads the node with the secret polynomial share $f_u(y) = f(i||Id_u,\ y)$.

Note that it is impossible that two different nodes can have the same secret polynomial share, so a node can never lie on its real identifier or the real generation number to which it belongs. Indeed, suppose that $u \in G_i$ and $v \in G_j$, with $i \neq j$. $f_u(y) = f_v(y)$ is possible, only and only if $i||Id_u = j||Id_v$. Each generation's number is exactly 2 bytes length, and each node identifier is exactly 4 bytes length, so $\mid i||Id_u \mid = \mid j||Id_v \mid =$ exactly 6 bytes. With our well formatted extended node identifier (2 bytes generation number, 4 bytes node ID), starting from an extended node identifier $i||Id_u$, it's impossible to find another distinct node identifier $j||Id_v$ where $i \neq j$ or $Id_u \neq Id_v$.

4.2 Pair-Wise Key Establishment Phase

Suppose that the BS deployed some previous generations, say the i first generations (1, 2, ..., i), and just deployed generation $i+1$. In our protocol, nodes know the highest deployed generation's number $i+1$ through a mechanism we describe in section 5.

Let $u \in G_j$ a newly deployed node. It is obvious that as a well-behaving node, $u \in G_{i+1}$. Node u tries to establish secure links with its direct neighbors by locally broadcasting a *Hello* message:

$$u \rightarrow * : \qquad Hello, j, Id_u, N_u$$

where N_u is used to guarantee response freshness. Let $v \in G_z$, where $z \leq i+1$, a neighbor node of u receiving its message. For node v to decide serving node u, node v follows two steps:

1. v checks if $j{=}i+1$, to verify whether u belongs to the newly deployed generation. If the verification fails, it simply rejects the request of node u, because u is normally already deployed.

2. If v verifies that $j{=}i+1$ then:
   - If v belongs to generation $z \leq i$, then v computes $K_{vu} = f_v(i+1||Id_u)$ and sends back to node u the following message:

$$v \rightarrow u : \qquad z, Id_v, N_v, MAC_{K_{vu}}(z, Id_v, N_v, N_u)$$

   - If $j{=}z{=}i{+}1$ ($u, v \in G_{i+1}$), then:
     - If the timer set by node v (to the value T_{est}, see section 3.3), did not expire, do the same treatment as the previous case.
     - If the timer expired, reject the request, because node u is suspected to be malicious.

Upon receiving node $v's$ message, node u computes $K_{uv} = f_u(z||Id_v)$, and checks the message authenticity. If the message is authenticated, node u sets

K_{uv} as the shared pair-wise key with v, and sends to v the following message to conclude and mutually authenticate the key establishment process:

$$u \rightarrow v: \qquad ok, MAC_{K_{uv}}(ok, N_v)$$

Upon receiving node $u's$ response, node v checks the message authenticity using K_{vu}, and, if successfully done, node v sets K_{vu} as the shared pair-wise key with u, otherwise (failed authentication, or non received response), it erases K_{vu}.

At the end of this phase, either a pair-wise key is established between two valid nodes, or the pair-wise key establishment fails in case one of the two nodes is suspected of being a clone or a false node. The described protocol guarantees that any served key establishment request, originates from a node belonging to the newly deployed generation $i+1$. However, the current version of the protocol fails to detect two particular attempts of false key establishment. The first attempt is when an attacker compromises a newly deployed node of generation $i+1$ and deploys clones in the neighborhood of nodes of older generations, and the second attempt is when an attacker compromises an older deployed node, and tries to respond to the *Hello* messages of the newly deployed nodes. Solutions to these two particular attempts are presented in section 7.

4.3 Key-Path Establishment Phase

In our scheme, two non neighboring nodes might attempt to establish a secret key. The two nodes might belong to the same generation, or to two different generations. Moreover, the node initiating the establishment might be from a higher generation, same generation, or lower generation than the target node. This raises a problem because we supposed that only newly deployed nodes are eligible for requesting key establishment. We rely on the previously established pair-wise keys in order to overcome this problem, and to guarantee each node that the other node is a valid node and not a cloned node or a false injected node.

Let $u \in G_i$ and $v \in G_j$ be two distant nodes that need to establish a secret key, and consider that u initiates the communication, and u knows Id_v and that $v \in G_j$. First, u must find a secure path to node v, formed by previously established secure links. Once the path is found, node u sends to v the following *Key Establishment Request* (KER) message:

$$u \rightarrow v: \qquad i, Id_u, N_u, MAC_{K_{uv}}(i, Id_u, N_u)$$

where $K_{uv} = f_u(j||Id_v)$. The KER message is sent in a secure path, where each node in the path, including u and v, authenticates the message with the key it shares with the previous node in the path. Upon receiving the KER message, node v computes $K_{vu} = f_v(i||Id_u)$, and then checks the authenticity of the message. If the KER message is authenticated, then v sends back to u the *Key Establishment Confirmation* (KEC) message:

$$v \rightarrow u: \qquad ok, MAC_{K_{vu}}(ok, N_u)$$

to conclude the key-path establishment. Node u checks the authenticity of the KEC message, and in case of unsuccessful authentication, it erases K_{uv}.

Note that thanks to the secure path, each node is ensured that the other node is neither a cloned node nor an injected node. Indeed, suppose that the path is going through nodes: u, w_1, ..., w_i, w_{i+1}, ..., w_r, v. According to section 4.2, each pair-wise key (secure link) in the network is established between two valid nodes, which are neither cloned nodes nor false injected nodes. As a consequence, the fact that u found a secure path and received the KEC message means that node v is a valid node, and the fact that node v received the KER message through the path, means that node u is a valid node.

5 Determining the Highest Deployed Generation

Now let describe how nodes know the number of the highest deployed generation, as seen in section 4.2.

The BS initially defines a static scheduling for generations deployment. The BS considers deploying the first generation G_1 at instant $T_1 = 0$, which serves also as a reference within deployed nodes for synchronization and time counting. If a period T is defined between each generation deployment, each node of generation i needs only be loaded with its generation's deployment time T_i, and the period time T.

After nodes deployment, when a node $u \in G_i$ asks for key establishment, a neighbor node v of an older generation j verifies that u is a node of the newly deployed generation G_i, by verifying that $0 < t_{current} - (i-1) * T < T$, where $t_{current}$ is the current time in node v. If this inequality is not verified, v rejects the request of u, because u is not a node from the highest deployed generation.

6 Computation and Memory Costs

Now, let consider the computation and memory costs of our protocol. For the memory cost, each node stores its extended identifier (generation number, node ID), its polynomial share and the established pair-wise keys. An extended identifier is 6 bytes length (see section 4.1). A polynomial share is represented by $t+1$ coefficients, plus the modulo Q. If we choose a modulo Q of 8 bytes, as in [11], and $t=100$, each node needs 816 bytes memory to store its polynomial share. In addition, each established pair-wise key needs 8 bytes of memory.

For the computation cost, each node needs to evaluate its polynomial share for each pair-wise key establishment. As described in [11], evaluating a polynomial share requires t modular multiplications and t modular additions in a finite field F_Q. However, because a sensor's CPU does not manipulate words of 64 bits (8 bytes), and the more powerful of them, like MOTEIV [12], handles words of 16-bit only (2 bytes), more modular multiplications and modular additions are needed. Consequently, in a 16-bit CPU processor, evaluating a t-degree polynomial share $f_u(y)$ over a finite field F_Q, where Q is a prime number of 64 bits, and y is an extended identifier (see section 4.1) of 48 bits (6 bytes), requires $4 \times t$ modular additions, and $12 + 28 \times (t - 1)$ modular multiplications.

7 Security Analysis of Our Solution

The proposed security analysis of our protocol focuses on the resilience to nodes compromising, and other features.

7.1 Resilience to Nodes Compromising

Now, let consider the resilience to nodes compromising, which refers to the capability of an attacker to inject cloned nodes and new nodes in the network, using the key materials it gets from the compromised nodes.

Injecting Nodes with False Ids. It is clear that as long as an attacker compromises fewer than $t + 1$ nodes, new nodes with non-existing Ids can not be injected in the network. In our protocol, each node u possesses a unique polynomial share bound to its identity $f_u = f(i||Id_u,\ y)$, where $u \in G_i$. After compromising node u, an attacker can not create node u', with a new identity $Id_{u'}$ and a new polynomial share $f_{u'}(y) = f(i||Id_{u'},\ y)$. In addition, an attacker can not use the polynomial share f_u, because node u' will fail to establish secret keys with this polynomial share using the new identity $Id_{u'}$.

As a conclusion, our protocol is resilient to the injection of false nodes with non-existing identifiers in the network.

Injecting Cloned Nodes. In our protocol, and under our assumptions of section 3, an attacker is highly unlikely to deploy cloned nodes, and convince his neighbors of the validity of the clones.

Table 2 summarizes the different scenarios for key establishment. Four possible cases of key establishment can be differentiated according to the generations to which the *requesting node* and the *responding node* belong. Next the behavior of the attacker is analyzed according to the role of the attacker which is either a requesting node, or a responding node.

First, let see how our protocol handles the two last cases $Old-New$, and $Old-Old$, where a node u of an older deployed generation asks for key establishment with a node of a newer generation, or an older generation. Remember that only nodes of the newly deployed generation are able to establish secure links with their neighbors. As a consequence, an attacker compromising an older deployed node $u \in G_i$, can not initiate key establishment with another deployed node $v \in G_j$ where $j > i$. In addition, the mechanism described in section 5 guarantees that all nodes of the network have the same view of the number of the highest deployed generation, so a node $u \in G_i$ of an older generation can not ask for key establishment with a node $v \in G_j$ where $j \leq i$.

Second, let see how our protocol handles the first case $New - New$, where an attacker compromises a newly deployed node and asks for key establishment with another newly deployed node of the same generation. By limiting the duration

Table 2. Identified scenarios for key establishment according to the generation of concerned nodes

Requesting node	Responding node
New	*New*
New	*Old*
Old	*New*
Old	*Old*

of key establishment phase for newly deployed nodes to T_{est}, even if an attacker compromises a newly deployed node in a period of time T_{comp} ($T_{comp} > T_{est}$), he cannot establish secure links with other nodes of the same generation, simply because the responding nodes will reject the request, as described in section 4.2.

Now let see how our protocol handles the second case, where a node of the newly deployed generation asks for key establishment with a node of an older generation. Again, two cases are distinguished:

- First, the newly deployed node (requesting node) is a cloned node of a compromised node that belongs to the newly deployed generation.
- Second, the responding node is a cloned node of a compromised node that belongs to an older generation.

For the first case, unfortunately the algorithm in section 4.2 does not handle this situation. This case is difficult to detect, because the cloned node looks like a node belonging to the highest deployed generation. One solution could be that an older deployed node accepts establishing secure links with nodes of any newly deployed generation only during a period of time T_{max} after the deployment of any new generation, where $T_{est} < T_{max} < T_{comp}$. According to section 5, nodes know the static scheduling of generations deployment, so each deployed node sets a timer to the value T_{max} when the time of deployment of a new generation is reached. Because an attacker needs at least a time T_{comp} in order to compromise a newly deployed node, we are practically ensured that an attacker compromising a newly deployed node, can not establish secure links with nodes of older generations, because these nodes will reject his request.

The second case is also difficult to detect, because the newly deployed node is asked for key establishment, and it has no way to check whether the responding node is a cloned node. The problem is even more difficult if the clone stays inactive or silent until a newly node is deployed. At this time, the clone might become active and establish a secure link with it by simply responding to its request. In this scenario, because the cloned node does not ask its neighbors for key establishment, it cannot be detected, so the newly deployed node cannot be prevented. One solution to this problem is that deployed nodes which are neighbors of both the newly deployed node and the cloned node, detect that a neighbor node exists but they have no secure links with it, so they conclude that the node is a malicious node. As a consequence, an informative message is sent by them to the newly deployed node which erases any established key with the cloned node.

7.2 Node Revocation and Intrusion Detection

As described above, in the four possible cases of key establishment, an active attack is always detected. Moreover, a silent attacker (intruder) is also detected when he tries to respond to key establishment requests from newly deployed nodes, and the newly nodes are then notified. As a consequence, the identity of the compromised node is known, so the neighboring nodes of the cloned node can either launch a distributed revocation against it, or notify the BS which broadcasts a revocation message in the network for revoking both the compromised node and its clones.

8 Related Works

Liu et al. [4] propose a distributed location-aware key establishment protocol, based on bivariate symmetric polynomials. The protocol assumes that the network is formed by simple nodes, and some sufficiently dedicated nodes called the service nodes, which are elected amongst sensors after deployment. These nodes play the role of *trusted* key servers in the network and are assumed to be *non-compromised*. The protocol also assumes that once nodes are deployed, they know their exact location coordinates (x, y). After deployment, each service node creates a distinct t-degree bivariate polynomial, and then securely initializes each neighbor node with its secret polynomial share, using the unique location coordinates of the node. The protocol is resistant and resilient to node compromising, as long as the service nodes are not compromised, and there are fewer than $t+1$ compromised nodes initialized by the same service node. However, if a service node is compromised - which is a current threat because a service node is just a non tamper resistant sensor node - an attacker can inject clones and new nodes with new positions, deploy them in the neighborhood of the compromised service node, and establish secure links with any nodes of the network.

Dutertre et al. [9] suppose that nodes are deployed in n successive generations, and can not be compromised in a period of time less than $T_{comp} > T_{est}$ (see section 3.1). Each generation is loaded with a unique two master keys, used respectively for authentication and key generation between the nodes of the generation. Once a deployed node successfully establishes secure links with its neighbors of the same generation, it erases these two keys to prevent from attacks. In order to establish secret keys between nodes of two different generations, each generation i is also loaded with a unique secret group key GK_i that enables nodes to establish secure links with previously deployed generations $j = 1, \ldots, i-1$. In addition, each node u of generation i is loaded with a unique random value R_u, and a secret key $Su_j = H(GK_j, R_u)$ for each future generation $j = i+1, \ldots, n$, allowing it to establish secure links with nodes of newly deployed generations.

The group key GK_i is also deleted at the end of the key establishment phase. The protocol is poorly resilient to nodes compromising, as an attacker compromising a node of generation i, can establish secure links with nodes of the future deployed generations, using the compromised secret keys Su_j. Moreover, if an attacker compromises a node u of generation i in a time period less than T_{est}, he might retrieve the master keys of generation i, and the group key GK_i. As a consequence, he can deduce secure links established between nodes of generation i, and can inject cloned nodes and false nodes anywhere in the network.

Bhuse et al. propose a key distribution protocol based on the use of the Hughes's variant of the DH protocol with encrypted key exchange (HDH-EKE) [13], and based on the assumption that nodes can not be compromised, and even if they are, then they self destroy without revealing their secret cryptographic materials. All nodes are initially loaded with a common password P used for authentication, and after deployment, nodes self organize into clusters, and elect one of them to act as a key server. The key server of each cluster generates a cluster key, and securely distributes the key to each one-hop neighbor using the HDH-EKE protocol, which in turn will pursue the distribution of the key to its neighbors in the same manner, until all nodes of the cluster posses the cluster key. The cluster key is used for encrypting messages and authenticating them inside the cluster. In order to send packets between two different clusters, boarding nodes, which posses the cluster keys of two or more clusters will act as a gateway by decrypting/encrypting messages from the source cluster to the target cluster. The key server periodically updates the cluster key, by sending a random counter value used along with the secret password, and the current cluster key to produce the new cluster key. The main problem of this protocol is its high computation overhead due to the use of modular exponentiations (public key cryptography), its weak authentication mechanism. The protocol is resilient against nodes compromising as long as an attacker cannot retrieve the secret common password P. However, it's unlikely that sensors can be tamper-resistant [10] [14], where memory containing the secret cryptographic materials is hardware-protected, because this will increase significantly the cost of sensors, and sensor nodes are intended to be very inexpensive.

9 Comparison with Previous Work

As we have seen in the description of some previous works done in the literature, most of them lack to provide resilience to nodes compromising, and those providing some degree of resilience rely on some assumptions, that can not be met easily. For essence, [6] and [4] suppose that nodes are tamper-resistant devices, so they can not be compromised or they self destroy when they detect that they are under attacks, and [4] relies also in the assumption that nodes know their locations coordinates, in-order to tie each node's secret key material (i.e. its secret polynomial share) to its location coordinates, so even if in attacker succeeds into compromising a node and creates some cloned nodes, it can not deploy them anywhere in the network. Someone can suppose that the future generation of sensors will be tamper-resistant. However, tamper-resistant devices are expensive, and constructors tendency may be to keep sensors at lower prices, with an increase of memory and computation capacities, instead of making them tamper-resistant. Localization in WSN is still under research, and the actual solutions assume that either nodes are equipped with GPS receivers, or the existence of some trusted nodes (at least three) on the perimeter of the network, that provide nodes with their locations coordinates. The first solution is unlikely to be deployed in sensors, and is energy consuming, and the second solution requires multiple trusted entities, and the resulting location coordinates are prone to errors. In our protocol, we don't assume that nodes are tamper-resistant devices, and we don't consider nodes locations.

Some other works like [5], [9] and [15], suppose that nodes share some common secret key(s) they use for key establishment, which will be erased from their memory straightforward after they finish key establishment when they are first deployed. These protocols suppose, as we do, that nodes can not be compromised in time less then T_{comp}, and that any newly deployed node needs at most a time $T_{est} < T_{comp}$ to establish keys with its neighbors. However, in the previous protocols, if an attacker succeeds to compromise a node in time less then T_{est}, it will have access to all its secret key materials, especially the common secret key(s), consequently the entire network security can be compromised, because established keys between non-compromised nodes can be retrieved, and future established keys can also be computed, and evidently cloned nodes and new nodes with non-existing identifiers can be easily injected. In our protocol, only if an attacker compromises a newly deployed node (which belongs to the newly deployed generation) in a time less than T_{est}, it will be able to deploy cloned nodes in the network and establish pair-wise keys with them, but the attacker can not compute any pair-wise key established in the network between two non-compromised nodes. If an attacker compromises an old deployed node (which belongs to an old deployed generation), it can not deploy cloned nodes of it, and even if the cloned nodes launch a silent attack (see section 7.2) they will be detected.

10 Conclusion

Our proposed solution improves considerably resilience to nodes compromising compared with other protocols, and does not require any prior knowledge relative to nodes deployment, and any common secret key pre-establishment between nodes. Moreover, the solution does not rely on non-realistic assumptions like supposing that compromised nodes do not divulge their secret keys or that some nodes in the network can not be compromised.

Our protocol uses t-degree polynomial-based key generation protocol for achieving resistance to nodes compromising, and the proposed group-based deployment scheme for resilience against nodes compromising, where only nodes of the newly deployed generation ask for key establishment. In addition, the proposed mechanism for determining the highest deployed generation, guarantee that nodes will respond only to the newly deployed nodes' requests, and limiting the duration of key establishment makes it practically impossible for an attacker to succeed in establishing keys in the network. Moreover, our protocol supports detection of silent attackers (intruders) and can be enhanced to achieve a distributed revocation. In a future work, we'll implement our protocol to evaluate its real resiliency to nodes compromising, and extend it with a distributed revocation mechanism.

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y.: Wireless sensor networks: a survey. Computer Networks 38 (2002) 393–422
2. Chan, H., Perrig, A., Song, D.: Random Key Predistibution Schemes for Sensor Networks. In IEEE Symposium on Security and Privacy. Okland California USA (2003) 197–213
3. Liu, D., Ning, P., Du, W.: Group-Based Key Pre-Distribution in Wireless Sensor Networks. In Proc. of the 4th ACM workshop on Wireless security. Cologne Germany (2005) 11–20
4. Liu, F., Major Rivera, J. M. , Cheng, X.: Location-aware Key Establishment in Wireless Sensor Networks. In Proc. of the International Conf. on Wireless Communications and Mobile Computing. Vancouver Canada (2006) 21–26.
5. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large Scale Distributed Sensor Networks. In Proc. of the 10th ACM Conf. on Computer and Communications Security. Washington DC USA (2003) 62–72
6. Bhuse V., Gupta A., Pidva R. (2003) A Distributed Approach to Security in Sensornets. IEEE Vehicular Technology Conference: 3010-3014.
7. Blundo, R., Suntis, A. D., Herzbeg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly secure key distribution for dynamic conferences. In Proc. of the 12th Annual International Cryptology Conference on Advances in Cryptology. Springer-verlag, UK (1992) 471–486
8. Yu, Z., Guan, Y.: A Robust Group-based Key Management Scheme for Wireless Sensor Networks. IEEE Wireless Communications and Networking Conference. New Orleans USA (2005) 1915–1920
9. Dutertre, B., Cheung, S. , Levy, J.: Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust. SDL Technical Report SRI-SDL-04-02, SRI International (2004)
10. Perrig, A., Szewczyk, R., Wen, V., Cullar, D., Tygar, J. D.:"Spins: Security protocols for sensor networks". In Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking. Rome Italy (2001) 189-199
11. Liu, D., Ning, P.: Establishing Pairwise Keys in Distributed Sensor Networks. In the Proc. of the 10th ACM Conference on Computer and Communication Security. Washington DC USA (2003) 52–61.
12. TmoteSky wireless sensor module. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf
13. DH Key-Excchange Protocols. https://www.cs.tcd.ie/courses/baict/bass/4ict11/Cou sewo k/4ICT11MT6.2.pdf
14. Karlof, C., Wagner, D.: "Secure routing in wireless sensors networks: attacks and countermeasures". Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols (2003)
15. Dimitriou T., Krontiris I. (2005) A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks. In Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium.

Efficient Use of Random Delays in Embedded Software

Michael Tunstall[1] and Olivier Benoit[2]

[1] Department of Electrical & Electronic Engineering,
University College Cork, Cork, Ireland
miket@eleceng.ucc.ie
[2] Gemalto, Security Labs,
Avenue des Jujubiers,
La Ciotat, F-13705, France
olivier.benoit@gemalto.com

Abstract. Random delays are commonly used as a countermeasure to hinder side channel analysis and fault attacks in embedded devices. This paper proposes a different manner of generating random delays, that increases the desynchronisation compared to random delays whose lengths are uniformly distributed. It is also shown that it is possible to reduce the time lost due to the inclusion of random delays, while maintaining the increased desynchronisation[1].

Keywords: Smart card security, fault attack countermeasures, side channel attack countermeasures.

1 Introduction

The use of random delays in embedded software is often proposed as a generic countermeasure against side channel analysis, such as Simple Power Analysis (SPA) and statistical analysis of the power consumption [6] or electromagnetic emanations [4]. Statistical analysis is meaningless in presence of desynchronisation; an attacker must resynchronise the acquisitions at the area of interest before being able to interpret what is happening at a given point in time.

This effect is discussed in detail in [3], where the case of hardware random delays is considered. This involves clock cycles being added into a process at random points to create desynchonisation. An attack based on taking the integral of adjacent points is proposed to find the information required to conduct a Differential Power Analysis (DPA) [6]. In this paper, the case where a software delay is introduced at various points in a process to introduce desynchonisation is discussed. This introduces a delay in the process that is too large to allow a similar attack to be conducted.

It has also been proposed as a countermeasure against fault attacks that require a high degree of precision in where a fault is injected [1]. Random delays will make it difficult to implement these attacks, as the target point in time is

[1] Work done while the first author was employed by Gemalto (patent pending).

constantly changing its position. An attacker is obliged to attack the same point numerous times and to wait until the fault and the target coincide.

In both cases, the greater the variation caused by the random delay, the more difficult it is to overcome. Nevertheless, the use of random delays is problematic, as it serves no purpose other than to desynchronise events. Therefore, it cannot prevent an attack; it just renders the attack more complex. This paper proposes an improvement to the distribution of the lengths of individual random delays, to improve the overall efficiency, i.e. the average time loss versus the work required to conduct side channel or fault attacks.

The paper is organised as follows. Section 2 describes how random delays are used in embedded software to defend against side channel analysis and fault attacks. Section 3 describes how the random value used to generate the lengths of these random delays can be modified, to have a useful effect on the cumulative distribution after several random delays have occurred. Section 4 describes an attack scenario and the amount of analysis that would be required to benefit from the modified distribution. This is followed by the conclusion in Section 5.

2 Software Random Delays

A software random delay is inserted into code to prevent an attacker from being able to determine what is happening at a specific moment in time during a command without some *a posteriori* analysis. In general, this will consist of a dummy loop where a random value is generated and then decremented until the random value reaches zero before executing any further code. An example of the sort of code that could be used is given below in the 8051 assembly language:

```
    mov   a, RND
    mov   r0, a
 Delay_Label:
    djnz  r0, Delay_Label
```

This adds very little extracode to the overall program as this moves the random value held in the register RND to another register (via the accumulator), and then decrements this value and loops until the accumulator contains zero. Generally, random number generators in embedded devices are based around a noisy resistor or a clock generator with a bad ring, and are tested using a suite of tests (e.g. [5]). The value placed into the accumulator can therefore be considered to be uniformly distributed across all the values it can take. In the case of 8051, an 8-bit architecture, this would be expected to be an integer from the interval $[0, 255]$.

These loops are inserted to prevent statistical power analysis (e.g. DPA [6]) and fault attacks [1]. An example of some power consumption acquisitions that include a random delay is shown in Figure 1. All three acquisitions are synchronous on the left hand side of the figure. This is followed by a random delay, visible due to the repeating pattern of the loop after which the acquisitions are no longer synchronous.

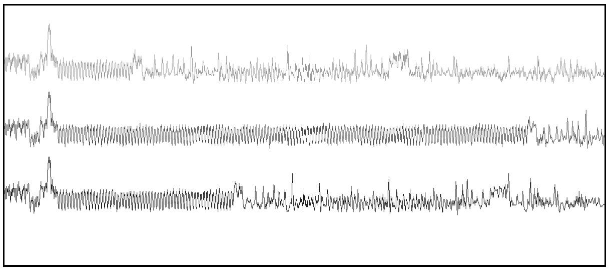


Fig. 1. The random delays visible in the power consumption over time

To conduct a statistical power analysis, an attacker needs to exploit the relationship between the data being manipulated at a given point in time and the power consumption. In order to do this, the acquisitions need to be synchronised *a posteriori* at the point that an attacker wishes to analyse. As the size of the random delay increases, an attacker is obliged to acquire more information to be sure of acquiring the point of interest. The amount of work required to resynchronise the acquisitions also increases, as pattern-matching software will have to search in a larger window to find the same point in each acquisition. It is therefore of interest to maximise the variation of the cumulative distribution of several random delays, as there are likely to be several random delays that occur before a potential target for statistical power analysis.

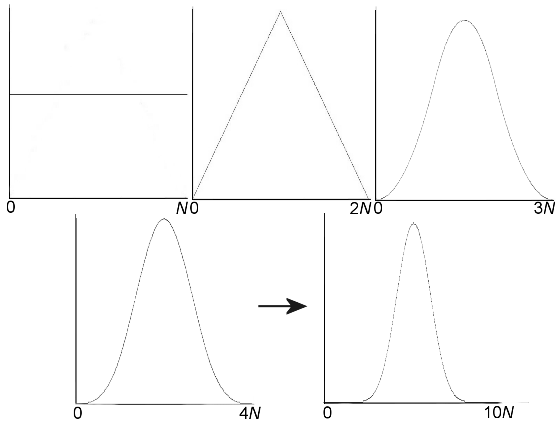
This is different to the case considered in [3]. The countermeasure evaluated in [3] involves introducing delays of one clock cycle but at random points in time. An attack against these small random delays can be conducted, based on evaluating the integral of a window of the acquisition rather than in a pointwise fashion. This is not applicable in the case of software random delays, as the information is spread over too many clock cycles.

An attacker wishing to conduct a fault attack needs to synchronise the fault injection with the event they wish to affect in real time. There are some features that can be used to synchronise with automatically, such as events on the I/O or an EEPROM write command, but these will only remove some of the synchronisation. In fact, it is considered prudent to include a random delay after such events that can take values from a comparatively large interval, to hinder an attacker that can synchronise in real time. An attacker is obliged to inject a fault where the event is likely to be and then repeatedly inject the fault until the event occurs at the desired point. This renders an attack more complex, as an attacker cannot be sure where the fault has been injected, and a means of detecting that the desired fault has occurred needs to be implemented (e.g. analysing the power consumption *a posteriori* [7]). This is another case where the variation at all target points needs to be maximised, as this will mean that an attacker has to conduct more attacks before being successful. A notable exception to this is the first published fault attack, that describes an attack against RSA when it is calculated using the Chinese Remainder Theorem [2]. In this case, the target is so large that the use of any form of random delay is unlikely to hinder the attack.

In general, an operating system will have a few large random delays placed at strategic points to prevent fault injection. Implementations of block ciphers will typically have smaller random delays placed between every subfunction of every round function (the use of random delays in public key cryptographic algorithms is somewhat similar, but is not considered in this article for clarity). More random delays are included in cryptographic algorithms predominantly to hinder attempts to conduct statistical power analysis. If an attacker synchronises a set of power acquisitions to analyse a given function, the following function will still be desynchronised and will require further work to synchronise.

This is a mild countermeasure, as it cannot prevent an attack from taking place. However, it can render an attack time consuming to the point where it is no longer practical.

Random delays are rarely used in one place, so an attacker is likely to be dealing with the cumulative delay of several random delays. The distribution of the cumulative delay is therefore distributed over the cumulative uniform distribution. This distribution is shown in Figure 2 for some small numbers of random delays, where the y-axis is normalised to show how the form of the distribution changes.



The cumulative random delay generated by a sequence of random delays generated from uniformly distributed random variables. The number of random delays considered are 1, 2, 3, 4, and 10, from top left to bottom right.

Fig. 2. The cumulative random delay

As can be seen, as the number of random delays increases, the distribution rapidly becomes binomial in nature, i.e. it approximates to a discrete normal distribution. It is also interesting to note that after 10 such random delays there is a tail on either side of the binomial, where the probability of a delay of this length occurring is negligible.

3 Modifying the Distribution

As described above, length of individual software random delays are generally governed by uniformly distributed random values. The distribution of the random value could be modified to improve the properties of the cumulative distribution. This section defines the criteria necessary for the modified distribution, and then describes how a solution was designed.

3.1 Design Criteria

The ideal criteria for the new distribution would be the following:

1. Random delays should provide more variation when compared to the same number of random delays generated from uniformly distributed random variables.
2. The overall decrease in performance due to the inclusion of random delays should be similar or better than if the random delays are generated from uniformly distributed random variables. This means that the mean of the cumulative distribution should be less than, or equal to, the mean of the cumulative uniform distribution.
3. Individual random delays should not make an attack trivial in the event that there is only one random delay between the synchronisation point and the target process.
4. It should be a non-trivial task to derive the distribution of the random delay used.

These conditions are the ideal; there will be some compromise needed between all of the conditions, as the uniform distribution will probably be preferable for criterion 3.

Our main aim in this paper is to optimise criteria 1 and 2, since the reverse engineering of a random delay distribution is rarely conducted (furthermore, it will be shown that this involves more effort than assuming the distribution is uniform) and individual efficiency is rarely applicable due to the numerous random delays used (as described above).

In order to be able to compare different distributions, the mean and the standard deviation are used to express the characteristics of the cumulative distribution of random delays. This was a natural choice as the cumulative distribution is based on a binomial distribution, which approximates to a discrete version of a normal distribution (characterised by the mean and variance). However, the standard deviation was chosen rather than the variance, as this represents the mean deviation from the mean.

3.2 Deriving a Suitable Distribution

In order to increase the variation in the cumulative distribution, the probabilities of the extreme values occurring were increased, i.e. the minimum and maximum values of one random delay. The formation of a binomial distribution after several random delays cannot be avoided, as the number of combinations close to the mean value is far too large. Any modification is also required to be subtle, as one random delay needs to be able to provide desynchronisation between two sensitive areas (criterion 2). It was initially assumed that this could be achieved by a distribution of the form shown in Figure 3.

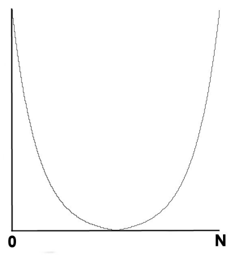


Fig. 3. An example of a modified probability function

To use this probability function in a constrained environment, for example a smart card, this needs to be expressed as a table where the number of entries for a given value represents the probability of that value being chosen. A uniformly distributed random value can then be used to select an entry from this table. The function that was chosen to govern the amount of entries for each value of x, where x can take integer values in the interval $[0, N]$, was:

$$y = \lceil ak^x + bk^{N-x} \rceil$$

where a and b represent the values of the probability function when x takes 0 and N respectively, and y governs the number of entries in a table for each value of x that can be used to represent this function. The sum of y for all values of x will therefore give the total number of table entries. k governs the slope of the curve and can take values in the interval $(0, 1)$. Different values for a and b are used so that a bias can be introduced into the sum distribution to lower the mean delay. The two elements ak^x and bk^{N-x} are both close to zero when $x \approx N/2$ for the majority of values of k that will be of interest. The ceiling function is therefore used to provide a minimum number of entries in the table for each value. This is important; if values are removed from the distribution it will decrease the amount of values the random delay can take and reduce its effect.

The parameters that generate this curve shape were changed and the effect on the sum distribution was tested to search for the best configuration that would satisfy the criteria given above. As given in the example, the length of each individual random delay can be an integer value in the interval $[0, 255]$.

For values of a and b that are approximately equal, the change in k will have an effect on the mean and the standard deviation as shown in Figure 4. This graph was generated by analysing a large number of random values generated by a random look-up on a table, as described above.

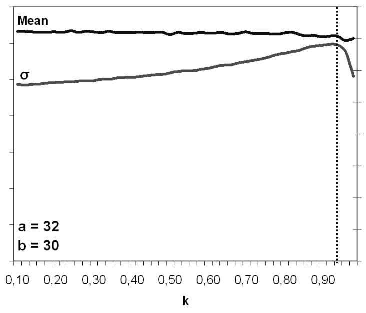


Fig. 4. The mean and the standard deviation against k for approximately equal values of a and b

Table 1. Parameter Characteristics for Tables of 2^9 Entries

a	b	k	Mean % decrease	σ % increase
22	13	0.88	13.4	34.5
23	12	0.88	16.3	33.5
23	15	0.87	11.6	35.4
24	11	0.88	19.7	32.1
24	14	0.87	13.4	34.9
25	10	0.88	22.6	30.7
26	6	0.89	32.8	23.5
26	9	0.88	25.6	29.0
26	12	0.87	19.7	32.5
32	8	0.86	31.4	25.8

The mean in Figure 4 is fairly constant for all the values of k tested. The variance shows an optimal value of $k = 0.92$. This experiment was repeated for various different values of a and b and the optimal value for k remains constant.

This value was used to examine the effect of varying b on the mean and the standard deviation, as the mean of the cumulative distribution can be reduced by introducing an asymmetry into the distribution of each individual random delay. The mean shows an almost linear relationship with b, and the standard deviation has a logarithmic relationship with b. The best configuration would be to minimise the mean value and maximise the standard deviation. This is not possible, and a compromise needs to be found between the two. $b = 16$ seemed to be a good compromise that was used to conduct further investigation.

To provide a table that can be efficiently implemented, the number of entries in the table should be a power of 2. A random number generator will provide a random word where the relevant number of bits can be masked off and used to read the value at the corresponding index the table to dictate the length of each the random delay. If the number of entries is not equal to a power of two any values generated between the number of entries and the next power of two

Table 2. Parameter Characteristics for Tables of 2^{10} Entries

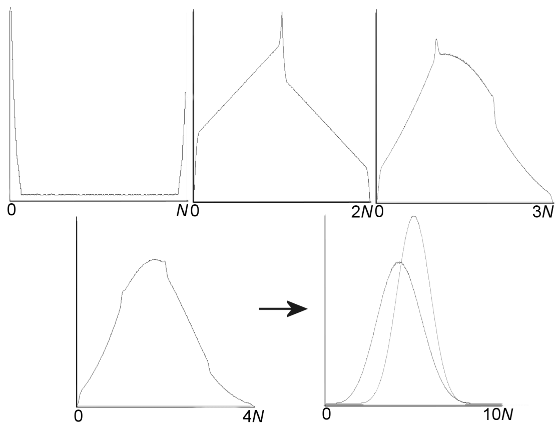
a	b	k	Mean % decrease	σ % increase
33	18	0.94	21.3	39.2
37	14	0.94	32.2	32.7
50	32	0.90	16.2	46.5
53	29	0.90	21.3	44.6
54	28	0.90	23.3	43.6
55	19	0.91	35.8	34.7
56	34	0.89	18.2	46.6
58	32	0.89	21.5	45.3
59	23	0.90	32.3	38.4
60	22	0.90	34.3	36.9

will have to be discarded. This testing of values will slow down the process and have a potentially undesirable effect on the distribution of the random delays, as suitable random numbers will only be generated with a certain probability.

The parameters that were found to naturally generate a table of 2^9 entries are shown in Table 1. It would be possible to choose some parameters and then modify the table so that it has 2^9 entries, but this was deemed overly complicated as this occurs naturally. The percentage changes shown are in comparison to the mean and standard deviation of a uniformly distributed random delay. The change in the mean and the standard deviation is not dependent on the number of random delays that are added together, which is not dependent on the number of random delays that occur.

As can be seen a large variation is visible in the change in the mean and standard deviation. It can also be seen that we cannot achieve the best standard deviation increase and mean decrease at the same time. The designer will have to make a compromise between maximising the standard deviation and minimising the mean.

Table 2 shows the parameters that naturally generate a table of 2^{10} entries. The effect of the modified random delays is more pronounced when based on a table of 2^{10} entries, as it is possible to approach the optimal value of 0.92 for k. However, it may not be realistic to have a table of 1024 in a constrained environment such as a smart card, although in modern smart cards the problem of lack of code memory is beginning to disappear. An example of the effect of using this sort of method to govern the length of random delays on the cumulative distribution is shown in Figure 5, where $a = 26$, $b = 12$ and $k = 0.87$. Again the y-axis is normalised to show the change in the cumulative distribution. The last graph shows the cumulative distribution for 10 random delays generated from random values from the modified distribution plotted alongside the cumulative distribution for 10 random delays generated from uniformly distributed random delays. The difference in the mean and standard deviation can be clearly be seen,



The cumulative random delay generated by a sequence of random delays generated from a biased distribution. The number of random delays considered are 1, 2, 3, 4, and 10, from top left to bottom right. The last graph also shows the distribution of 10 uniformly distributed random delays for comparative purposes.

Fig. 5. The cumulative random delay using a modified distribution

and the tail present on the left hand side is much smaller for the cumulative distribution of random delays governed by random values from the modified distribution.

This distribution satisfies the criteria 1 and 2 as described in the introduction. It is unlikely to satisfy criterion 3 as the distribution does have some undesirable properties. The probability of a 0 being produced by the table is $(a+1)/T$, where T is the number of entries in the table.

4 Reverse Engineering the Distribution

If an attacker knows the distribution of the random delays used, this information could potentially be used to increase the speed of an attack rather then hinder an attack. However, the situations where this information is useful are rare. If one random delay can be isolated, attacks can be designed around the fact that the distribution of the length of the random delay has been modified. The aim of changing the distribution is to make an attacker work harder to conduct a side channel or fault attack. Situations where the modification allows an attacker to increase the efficiency of an attack are highly undesirable.

4.1 Potential Attack Scenarios

In the case of statistical power analysis, an attacker is unlikely to be able to provoke a situation where the desynchronisation present at a given point is caused by one random delay. A suitable target for statistical power analysis will only occur after several functions have already taken place within a cryptographic algorithm and, therefore, after several random delays (as described in Section 2). The same argument can be applied to faults injected within a cryptographic algorithm.

If an attacker synchronises a set of power consumptions at a given point in a cryptographic algorithm, then there will only be one random delay between that point and the following function. There will be a certain number of acquisitions that remain synchronised because of the modified distribution. In theory, an attacker could take $a + 1$ times as many acquisitions than would normally be necessary to conduct a statistical power analysis, and then extract the curves that remain synchronised to have a large area synchronised at the desired point. However, this represents much more work than would be required to synchronise before and after the random delay in question.

The only point where an attacker is able to isolate one random delay is when attempting to inject a fault into the operating system running on an embedded device. In Section 2 the mechanisms for synchronising with events in the operating system are described, followed by one random delay that hinders attacks on a large area of the operating system. If a modified distribution is used in this situation, an attacker could greatly increase the chances of injecting a fault at a desired point by attacking the point in time that would correspond to the minimum value of the random delay. It is therefore of interest to know how to determine the distribution of a random delay at a given point in a command.

4.2 Hypothesis Testing

In order to determine whether the length of a random delay at a certain point in a command is uniformly distributed or not, a reasonable amount of data would need to be collected and the lengths of the random delays stored. These values can then be tested statistically to determine whether or not they correspond to a uniform distribution or not. This can be done by conducting a χ^2 test on the acquired data. An attacker will most likely be required to measure each delay by hand, which will be a lengthy and tedious process. It would be possible to develop a tool for a given chip that would generate this information, but as soon as the chip is changed the tool would need to be updated as different chips change the power consumption in different ways.

This process can be quite complex if hardware random delays are also present. These normally take the form of randomly inserted clock cycles (where the chip randomly insert clock cycles where no processing occurs) or an unstable internal clock. This is likely to add serious complexity to the resynchronisation process. These effects are ignored in this analysis for simplicity, but would make an attacker work much harder for the desired information.

It can be shown that the random delay being observed is not based on a uniformly distributed random value by using a χ^2 test with a null hypothesis that the random delays are uniformly distributed. In order to conduct this test the minimum frequency threshold should be around $5(N+1)$ (a rule of thumb given in [5]). In the example, given the minimum frequency threshold, 1280 samples would be required. In practice, far fewer samples are required to consistently provide evidence against the null hypothesis when one of the proposed distributions is used. However, the attacker cannot trust these results without repeating the test with independent acquisitions, so the actual amount of data required will probably still be around $5(N+1)$.

However, an attacker can base some attacks on the assumption that the length of a random delay may use the modified distribution, and not reduce the chances of the attack being successful. For example, an attacker injecting a fault after one random delay could attack the point in time corresponding to the minimum length of one random delay, thus maximising the chances of the attack being successful. If the length of a single random delay is uniformly distributed this strategy is as good as any other, as the chances of the minimum length occurring are the same as any other length.

5 Conclusion

In this paper we have demonstrated that the standard deviation of the length of cumulated random delays can be improved upon, by using a specific distribution for each individual random delay. It has also been shown that the expected amount of time lost due to random delays can be reduced to minimise the impact of this countermeasure on the performance of functions they are protecting. This is not presented as an optimal solution, but is assumed to be close to optimal.

The modified distribution presented satisfies all the design criteria given, except when one random delay is used after a synchronisation point to hinder attacks on the operating system, as described in Section 4.1. The strategy for implementing this countermeasure is therefore to use the uniform distribution where one random delay is being used to protect the operating system, and to use the modified distribution where numerous random delays are going to be used e.g. in cryptographic algorithms.

This represents an unusual situation, where a method of increasing the security of a process can also reduce the amount of computational time required. Given the nature of random delays, this increase in efficiency is actually a reduction in the time lost due to the use of random delays. Normally, in smart card implementations of cryptographic algorithms the addition of more secure features always comes with a reduction in performance.

References

1. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerers apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.
2. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking computations. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.

3. C. Clavier, J.-S. Coron, and N. Dabbous. Differential power analysis in the presence of hardware countermeasures. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer-Verlag, 2000.
4. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, 2001.
5. D. Knuth. *The Art of Computer Programming*, volume 2, Seminumerical Algorithms. Addison–Wesley, third edition, 2001.
6. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
7. D. Naccache, P. Q. Nguyen, M. Tunstall, and C. Whelan. Experimenting with faults, lattices and the DSA. In S. Vaudenay, editor, *Public Key Cryptography — PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 16–28. Springer-Verlag, 2005.

Enhanced Doubling Attacks on Signed-All-Bits Set Recoding

HeeSeok Kim[1], Tae Hyun Kim[1], Jeong Choon Ryoo[1], Dong-Guk Han[2,⋆], Ho Won Kim[2], and Jongin Lim[1]

[1] Graduate School of Information Management and Security, Korea University, Korea
`{heeseokkim, thkim, jcryoo, jilim}@cist.korea.ac.kr`
[2] Electronics and Telecommunications Research Institute(ETRI), Korea
`{christa, khw}@etri.re.kr`

Abstract. In cryptographic devices like a smart card whose computing ability and memory are limited, cryptographic algorithms should be performed efficiently. However, the issue of efficiency sometimes raises vulnerabilities against side channel attacks (SCAs). In elliptic curve cryptosystems, one of main operations is the scalar multiplication. Thus it must be constructed in safety against SCAs. Recently, Hedabou et al. proposed a signed-all-bits set (sABS) recoding as simple power analysis countermeasure, which is also secure against doubling attack (DA). In this paper we propose enhanced doubling attacks which break Hedabou's countermeasure based on sABS recoding, and then show the statistical approach of noise reduction to experiment on the proposed attacks in actuality. We also introduce a countermeasure based on a projective coordinate.

Keywords: Side Channel Attacks, sABS recoding, SPA-based analysis, scalar multiplication, Doubling Attack.

1 Introduction

Many designers of cryptosystems have proposed cryptographic algorithms based on theoretical security such as integer factoring problem and discrete logarithm problem. Even though these algorithms are proved as safe with mathematical tools, they could be vulnerable to physical attacks using additional information via side channel. Such type of attacks is referred to as Side Channel Attacks (SCAs) first introduced by Kocher [10]. In categories of SCAs, actively researched power analysis attack classifies into the simple power analysis (SPA) and the differential power analysis (DPA). To resist SPA among the power analysis attack, many researchers have proposed various countermeasures. Above all, two famous countermeasures are Coron's method [4] adding dummy operations and the scalar multiplication algorithm using singed all-bits set (sABS) recoding [7]. But, Coron's dummy method exposes a weakness by doubling attack (DA)

⋆ Corresponding author.

[5] introduced by Fouque et al. in 2003. Contrary to Coron's dummy method, sABS recoding based countermeasure is secure against doubling attack.

In this paper we propose two enhanced doubling attacks applicable to scalar multiplication algorithm based on sABS recoding, and introduce an experimental method to justify the practicality of the proposed attacks including experiment results. The one proposed attack is called recursive attack which is an analysis method finding the secret key from the most significant bit to the least significant bit in sequence through an adjustment of two input data to do the same elliptic curve doubling (ECDBL) in the vicinity of guessing secret bit. The other is called initializing attack which is an analysis method adjusting one input data to do fixed ECDBL at the guessing secret bit. We also propose a solution to detect equality of the compared two ECDBLs power signals by using a statistical approach of noise reduction when the noise is more important. Furthermore, we find that if we use a projective coordinate system to represent an elliptic curve element then it is secure against not only general doubling attack and but the proposed enhanced doubling attacks.

The remainder of this article is organized as follows. Section 2 represents that Coron's dummy method as an SPA countermeasure is vulnerable to SPA-based DA. Our proposed attacks are introduced in Section 3. These new attacks are SPA-based analysis methods applicable to scalar multiplication algorithm using sABS recoding among SPA countermeasures, and a practical attack method and a realistic possibility is showed in Section 4. Section 5 represents countermeasures on our attacks. Finally we conclude in Section 6.

2 Side Channel Attacks and Countermeasures

Since side channel attacks using additional information via side channel were introduced by Kocher, a various attack methods of this class have been proposed. There are fault insertion attack [2,20], timing attack [10], power analysis attack [11,12], electromagnetic emission attack [18], and so on. In power analysis attack, there are SPA that can expose secret key to be used by means of simple observation of a power consumption trace and DPA that analyzes multiple signals statistically without physical transformation of a smart card. DPA requires measuring a lot of power consumption and additional information such as description of implementation. But SPA is so simple. In this section, we introduce SPA and SPA countermeasures. Also we represent that Coron's dummy method known to be immune to SPA exposes a weakness against SPA-based DA.

2.1 Simple Power Analysis to ECC

Koblitz and Miller proposed elliptic curve cryptosystems (ECCs) in 1985 [9,14]. Because of short length of the secret key for guaranteeing the same security with RSA, ECCs are suitable for mobile devices such as mobile phones, smart cards, and PDAs which are limited at storage space and bandwidth. While cryptosystems such as RSA [16] use the operation of modular exponentiation, ECCs use the operation of scalar multiplication that is regarded as similar method. And so this operation is the most dominant operation in ECCs. Scalar multiplication is to compute dP from a point P on an elliptic curve. Algorithm 1 that is a standard method for computing the scalar multiplication works by scanning the secret key from MSB to LSB.

Algorithm 1. Double-and-add algorithm

```
Input : A point $P$, and $d = (d_{n-1}d_{n-2}...d_1d_0)_2$ , $d_{n-1} = 1$
Output : $dP$
    1. $S = P$
    2. For $i = n - 2$ downto 0
        2.1 $S = 2S$
        2.2 If $d_i = 1$, $S = S + P$
    3. Return($S$)
```

In Algorithm 1, the scalar multiplication for secret key d is carried out by scanning from MSB to LSB. If a specific bit of d is 1, the algorithm comes into operation of Step 2.1, 2.2. If not, that comes into operation of Step 2.1 only. In other words, depending on the key bit value, one carries out both elliptic curve addition (ECADD) and ECDBL, the other carries out ECDBL only. In general, ECADD has different power consumption from ECDBL [3]. Thus we can deduce the secret key by a power consumption of the scalar multiplication. This method that can expose a portion of secret key using only one signal is called by SPA.

2.2 SPA Countermeasures

Algorithms that have a conditional branch depending on the secret key are weak against SPA. For eliminating this weakness, algorithms that carry out unnecessary ECADDs regardless of the value of bit have been proposed.

SPA Countermeasure 1 - Dummy Operation. Algorithm 2 proposed as SPA countermeasure executes dummy operation when the value of bit is ‘0’.

Algorithm 2. Coron’s dummy method

```
Input : A point $P$, and $d = (d_{n-1}d_{n-2}...d_1d_0)_2$ , $d_{n-1} = 1$
Output : $dP$
    1. $S[0] = P$
    2. For $i = n - 2$ downto 0
        2.1 $S[0] = 2S[0]$
        2.2 $S[1] = S[0] + P$
        2.3 $S[0] = S[d_i]$
    3. Return($S[0]$)
```

This algorithm executes Step 2.2 at every loop regardless of the value of bit. In any loop that be carried out when the value of bit is 0, Step 2.2 is the superfluous ECADD operation. But in spite of this demerit in sense of efficiency, this algorithm is secure against SPA because it always compute both ECADD and ECDBL independent of a bit of the secret key. But this algorithm has problem for efficiency, moreover we will describe that that is insecure against SPA-based DA in the next subsection.

SPA Countermeasure 2 - sABS recoding. Other countermeasures are to change the binary representation of the secret key using signed digits. Among those countermeasures, sABS recoding method proposed by Hedabou et al. recodes the secret key into a new representation without zero bits by converting 00...01 into $1\bar{1}...\bar{1}\bar{1}$ where $\bar{1}$ means -1.

Algorithm 3 executes the scalar multiplication with this recoded representation. If the secret key is even, this algorithm carries out the operation of $dP = (d+1)P - P = t'P - P$ like Step 1 and Step 5 where t' is the sABS recoded value of $d+1$. Because this recoded value of the secret key does not have '0' bit, the sABS recoding method is secure against SPA: in Step 4 of Algorithm 3, ECADD or elliptic curve subtraction (ECSUB) whose power consumption is similar to ECADD is always carried out in every loop, and so this method does not come out the weakness against SPA. Note that we show that sABS recoding is secure against the original DA at the following section.

Algorithm 3. Scalar multiplication with sABS recoding

```
Input : A point $P$, and $d = (d_{n-1}d_{n-2}...d_1d_0)_2$ , $d_{n-1} = 1$
Output : $dP$
    1. if($d$ is even) then $t = d + 1$
    2. sABS recoded value of $t$ : $t' = (t'_{n-1}t'_{n-2}...t'_1t'_0)_2$, $t'_i \in \{-1, 1\}$
    3. $S = P$
    4. For $i = n - 2$ downto 0
        4.1 $S = 2S$
        4.2 if($t'_i = 1$) then $S = S + P$, else then $S = S - P$
    5. if($d$ is even) then $S = S - P$
    6. Return($S$)
```

2.3 Doubling Attack

Algorithm 2 exposes a weakness against DA that uses not DPA but SPA. Because DA is a SPA-based analysis, this attack method is much simpler than the existing DPA method.

Doubling Attack and Weakness of Coron's Dummy Method. DA is a possible method when an attacker has an ability that if the card computes

ECDBL(A) and ECDBL(B), he is able to check whether $A = B$ or not, even so actual values of A and B don't be recovered by himself. The basic idea of this attack is like the table below:

Input	1	**0**	1	**0**	**0**	1	**0**	**0**	1
P	0	$2P$	**4P**	$10P$	**20P**	**40P**	$82P$	**164P**	**328P**
	P	$3P$	$5P$	$11P$	$21P$	$41P$	$83P$	$165P$	$329P$
$2P$	0	**4P**	8P	**20P**	**40P**	80P	**164P**	**328P**	656P
	$2P$	$6P$	$10P$	$22P$	$42P$	$82P$	$166P$	$330P$	$658P$

We compare power signals when the card computes dP and $d(2P)$ for input point P and $2P$, this recovers all bits of the secret key through confirmation of equal power consumption by same ECDBL in the vicinity of the bit value '0'.

Security of sABS Recoding Against Doubling Attack. DA is the attack method using weakness that a certain bit of d is '0'. Hence, the original DA cannot be applied to the scalar multiplication with sABS recoding that does not have bit value '0'. In Algorithm 3, if an attacker tries DA to expect the value of t'_{n-l} that is the upper l-th bit of recoded value of the secret key (d or $d+1$), the values computed until the upper l-th bit for input point P and the upper $(l-1)$-th bit for input point $2P$ should be the same as the following equation.

$$(\sum_{i=0}^{l-1} t'_{n-l+i}2^i)P = (\sum_{i=0}^{l-2} t'_{n-l+i+1}2^i)2P \quad \Rightarrow \quad (\sum_{j=0}^{l-1} t'_{n-l+j}2^j)P = (\sum_{j=1}^{l-1} t'_{n-l+j}2^j)P. \quad (1)$$

Therefore, to satisfy equation (1), we know easily that $t'_{n-l} = 0$. However, because sABS recoded value is composed of '1' and '-1', this is not vulnerable to DA.

3 Proposed Attacks

Our paper proposes two attacks, recursive attack and initializing attack, against SPA countermeasure that executes the scalar multiplication using sABS recoding. Like original DA, these new two attacks are possible when an attacker has ability to decide whether $A = B$ or not when smart card computes ECDBL(A) and ECDBL(B). Our paper also offers an authenticity of this assumption through experimental result and theory in the next section. At first, we introduce our new attacks in this section.

3.1 Recursive Attack

In the proposed attack methods, recursive attack's basic idea is like follows: Suppose an attacker guesses a specific bit of the target secret key d, and he regulates two input values to have equal power consumption by the same ECDBL in the vicinity of the target bit. In this way, all bits of the secret key d can be discovered in sequence.

Input Value Regulation. Suppose that an attacker knows upper bits $t'_{n-1}t'_{n-2}\ldots t'_{u+2}t'_{u+1}$ of $t' = (t'_{n-1}t'_{n-2}\ldots t'_1 t'_0)_2, t'_i \in \{-1, 1\}$, which is a recoded value of secret key d in Algorithm 3, let us regulate input values to find the value of t'_u. When we get two input values of xP and yP, we should regulate values of x and y for originating same ECDBL in phase of operation from $i = u$ to $i = u-1$ for xP and from $i = u+1$ to $i = u$ for yP in Step 4.1 of Algorithm 3. If we guess the value of t'_u as 1, the value of S until $i = u$ for input value xP is $S = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u} + 1)xP$, and the value of S until $i = u + 1$ for input value yP is $S = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})yP$ in Step 4 of Algorithm 3. To originate the same ECDBL at this moment, if we select xP and yP satisfying this equation $(\sum_{i=u+1}^{n-1} t'_i 2^{i-u} + 1)xP = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})yP$, we can get the following values.

$$xP = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})P, \quad yP = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u} + 1)P \tag{2}$$

If we guess the value of t'_u as -1, xP and yP are the following values for the same reason as mentioned above.

$$xP = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})P, \quad yP = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u} - 1)P \tag{3}$$

$\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1}$ in equation (2) and (3) is a upper portion of t'_u that we are trying to find the bit of recoded value t' in Algorithm 3. Hence, if we name this value as k, two selected input values are kP and $(2k + 1)P$ $(kP, (2k - 1)P)$ in the case that we guess the value of t'_u as 1 (-1). In this way, we can find all bits of the secret key d from MSB to LSB in sequence.

Scenario and Example of Recursive Attack. In this section, we introduce the scenario of recursive attack for finding the entire value of the secret key, and then give a simple example to help understanding of this attack. Table 1 is the scenario of recursive attack for finding the entire information of the secret key.

Let us Consider this scenario. For example, if the secret key d is $(101010011)_2$ in Algorithm 3, then the value of t' becomes $11\bar{1}1\bar{1}1\bar{1}\bar{1}1$. In Table 1, suppose that the attacker already knows upper four bits of t' (upper 4 bits values : $11\bar{1}1 = (11)_{10}$). For now, he attempts to guess upper 5-th bit as 1. Hence, input values to know this bit are $11P$ used already to know the upper 4-th bit and $(2*11+1)P$ viewed in Step 4 of the scenario. And then he confirms as the table below whether the same ECDBL is originated in the vicinity of the upper 5-th bit or not.

Input	1	1	$\bar{1}$	1	$\bar{\mathbf{1}}$	1	$\bar{1}$	$\bar{1}$	1
$11P$	0	$22P$	$66P$	$110P$	$242P$	**462P**	$\cdots$	$\cdots$	$\cdots$
	$11P$	$33P$	$55P$	$121P$	**231P**	$473P$	$\cdots$	$\cdots$	$\cdots$
$23P$	0	$46P$	$138P$	$230P$	**506P**	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	$23P$	$69P$	$115P$	**253P**	$483P$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table 1. The Scenario of Recursive Attack

```
Step 1. Set $k = 1, i = 2$.
Step 2. Measure a power consumption $C_1$ related with the input point $P$.
Step 3. If $i = n$, goto Step 7.
Step 4. Measure a power consumption $C_2$ related with the input point $(2k + 1)P$.
Step 5. If $C_1$ and $C_2$ have the same ECDBL signal in the vicinity of the upper
            $i$-th bit, then $k = 2k + 1$.
        Else then,
            a. Measure a power consumption $C_2$ related with input point $(2k - 1)P$.
            b. $k = 2k - 1$.
Step 6. $C_1 \longleftarrow C_2$ and $i = i + 1$ goto Step 3.
Step 7. If the output about the input point $P$ is $(2k + 1)P$, then return $2k + 1$.
        Else then, return $2k - 1$.
```

In Step 5 of Table 1, because the attacker cannot confirm the same ECDBL signal in the vicinity of the top 5-th bit, he is able to decide this bit as -1. Hence, he measures the power signal C_2 about input point $21P = (2 * 11 - 1)P$. This power signal uses to know the upper 6-th bit, the next bit, of the secret key. For knowing the upper 6-th bit, the attacker guesses this bit as 1 and so he gets the power signal about input point $43P = (2 * 21 + 1)P$ in Step 4 of scenario. If this signal is compared with the signal C_2, it is as follows.

Input	1	1	$\bar{1}$	1	$\bar{1}$	**1**	$\bar{1}$	$\bar{1}$	1
$21P$	0	$42P$	$126P$	$210P$	$462P$	$882P$	**1806P**	$\cdots$	$\cdots$
	$21P$	$63P$	$105P$	$231P$	$441P$	**903P**	$1785P$	$\cdots$	$\cdots$
$43P$	0	$86P$	$258P$	$430P$	$946P$	**1806P**	$\cdots$	$\cdots$	$\cdots$
	$43P$	$129P$	$215P$	$473P$	**903P**	$1849P$	$\cdots$	$\cdots$	$\cdots$

Because the same ECDBL happens in the vicinity of the upper 6-th bit, the attacker can find the 6-th bit as 1. Through this mechanism, we can find the secret key d used in Algorithm 3 from MSB to LSB in sequence.

3.2 Initializing Attack

When recursive attack against sABS recoding recovers the secret key d from MSB to LSB in sequence, one bit of the secret key is exposed through inserting input value in the card at the minimum of one time and at the maximum of two times. We propose initializing attack as the second attack method against sABS recoding. This attack uses only one input value for recovering a bit of the secret key compared with recursive attack which uses 1.5 input value on the average. This method uses vulnerability that an attacker, with ease, can get an information of ECDBL from P to $2P$ for input value P in Step 4.1 of Algorithm 3.

Input Value Regulation. Initializing attack, similar to recursive attack, is an attack method to break the next bit in sequence when it knows upper certain bits of the secret key d in advance. The basic idea of this attack is that an attacker selects an input value xP such that the intermediate value computed up to a guessing bit of the key always becomes P and so originates ECDBL operation from P to $2P$ at the computation time of the next bit. If we suppose that the attacker knows upper bits $t'_{n-1}t'_{n-2}...t'_{u+2}t'_{u+1}$ of sABS recoded value t' in Algorithm 3, let us consider him to anticipate t'_u as 1. If his guess is right, computed value of S until the moment of $i = u$, in Step 4 of Algorithm 3, about input value xP is $S = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u} + 1)xP$. Hence, he will select the value x such that $S = P$. If $k = \sum_{i=u+1}^{n-1} t'_i 2^{i-u-1}$, the value x is given by

$$x = (2k+1)^{-1} \ mod \ (\sharp E).$$

Here, $\sharp E$ represents order of elliptic curve in ECCs.

The order of an elliptic curve is in the form of $q,\ 2q,\ 4q,\ 6q(q : prime)$ in standard documents ANSI X9.62 [1], FIPS 186-2 [15], SECG [17], WTLS [19], and ISO/IEC 15946-4 [8]. Because $\gcd(2k+1, 6q) = 3$ is possible in spite of $\gcd(2k+1, 2) = 1$ and $\gcd(2k+1, q) = 1$, there is the case that $(2k+1)^{-1}$ does not exist in the case of order $6q$. But, in case of $2k+1 = 3t$, $gcd(2k-1, 6q) = 1$ is satisfied because $2k-1 = 3t-2$ is not multiple of 3. Accordingly, in case of $gcd(2k+1, \sharp E) \neq 1$, the attacker guesses that t'_u is -1 not 1. The computed value S up to $i = u$ is $S = (\sum_{i=u+1}^{n-1} t'_i 2^{i-u} - 1)xP$ when the input value is xP in the Step 4, and so the attacker selects value x such that $S = P$. If $k = \sum_{i=u+1}^{n-1} t'_i 2^{i-u-1}$, the value of x is given by

$$x = (2k-1)^{-1} \ mod \ (\sharp E).$$

Our attack seems to be Goubin's Refined Power-analysis Attack [6], but ours uses the discriminative method that compares two waveforms (The method is introduced in Section 4). Also, while Goubin's attack can use only "special point" with zero coordinate, our attack has a merit that can use almost every points over elliptic curve.

Scenario and Example of Initializing Attack. In this section, we introduce the scenario of initializing attack for finding the entire value of the secret key, and then give a simple example to help understanding of this attack. Table 2 is the scenario of initializing attack for finding the entire value of the secret key.

Let us consider this scenario. For example, if the order of an elliptic curve is 73 and secret key d is $(101010011)_2$ in Algorithm 3, the value of t' becomes $11\bar{1}1\bar{1}1\bar{1}\bar{1}1$. In Table 2, suppose that the attacker already knows upper four bits of the secret key(upper 4 bits values : $11\bar{1}1 = (11)_{10}$). For now, he guesses the upper 5-th bit as 1 like Step 3 of scenario. Hence, the used input value to know this bit is $54P$ in Step 4 ($(2*11+1)^{-1}\ mod\ 73 = 54$). The attacker ascertains as the table below whether the card performs ECDBL of point P in the upper 5-th bit or not.

Table 2. The Scenario of Initializing Attack

```
Step 1. Set $k = 1,\ i = 2$.
Step 2. If $i = n$, goto Step 8.
Step 3. If $(2k+1)^{-1}\ mod\ (\sharp E)$ exists, then $k' = 2k+1$, $s = 1$.
        Else then, $k' = 2k-1$, $s = -1$.
Step 4. Compute $k = k'^{-1}\ mod\ (\sharp E)$.
Step 5. Measure a power consumption $C$ related with the input point $kP$.
Step 6. If $C$ have an ECDBL signal from $P$ to $2P$ in the upper $i$-th bit, then $k = k'$.
        Else if $s = 1$, then $k = k' - 2$.
        Else then, $k = k' + 2$.
Step 7. $i = i + 1$, goto Step 2.
Step 8. If the output about the input point $P$ is $(2k+1)P$, then return $2k+1$.
        Else then, return $2k-1$.
```

Input	1	1	$\bar{1}$	1	$\mathbf{\bar{1}}$	1	$\bar{1}$	$\bar{1}$	1
$54P$	0	$35P$	$67P$	$26P$	$14P$	**66P**	$\cdots$	$\cdots$	$\cdots$
	$54P$	$70P$	$13P$	$7P$	**33P**	$47P$	$\cdots$	$\cdots$	$\cdots$

In Step 6 of scenario, because the attacker cannot find ECDBL signal of point P in the upper 5-th bit, he is able to know this bit as -1 and so set $k = 23 - 2$. For knowing the upper 6-th bit recursively the attacker guesses this bit as 1, and so if he gets a power signal about input point $17P = (2 * 21 + 1)^{-1}P$ in Step 3 and Step 4, it is as follows.

Input	1	1	$\bar{1}$	1	$\bar{1}$	**1**	$\bar{1}$	$\bar{1}$	1
$17P$	0	$34P$	$29P$	$24P$	$9P$	$57P$	**2P**	$\cdots$	$\cdots$
	$17P$	$51P$	$12P$	$41P$	65	**P**	$58P$	$\cdots$	$\cdots$

Because the card performs ECDBL of point P in the upper 6-th bit, the attacker can know this bit as 1. Through this mechanism, attacker can find the entire value of the secret key d in sequence.

4 Statistical Approach of Noise Reduction

Both the proposed attacks and DA are accomplished if an attacker is able to become aware whether the smart card computes ECDBL of the same point or not through two ECDBL signals only. In this section, we propose that the above assumption can accomplish in actuality using experimental results. For example, we show that how to an attacker knowing the upper $(i-1)$ bits of the secret key detects the upper i-th bit in the recursive attack under assumption that he/she can distinguish power signals between ECDBL and ECADD.

Let $D_{i,1}^{(j)}$ ($D_{i,2}^{(j)}$) be a j-th ECDBL signal related with the first (second) input point $P_{i,1}$ ($P_{i,2}$) for knowing i-th bit. Also, $A_{i,1}^{(j)}$ ($A_{i,2}^{(j)}$) denotes a j-th ECADD signal related with the first (second) input point $P_{i,1}$ ($P_{i,2}$) for knowing i-th bit. Let the power signals related with input points $P_{i,1}$ and $P_{i,2}$ be represented as

$$P_{i,1} \Rightarrow D_{i,1}^{(1)} A_{i,1}^{(1)} D_{i,1}^{(2)} A_{i,1}^{(2)} D_{i,1}^{(3)} A_{i,1}^{(3)} D_{i,1}^{(4)} A_{i,1}^{(4)} D_{i,1}^{(5)} A_{i,1}^{(5)} \cdots,$$
$$P_{i,2} \Rightarrow D_{i,2}^{(1)} A_{i,2}^{(1)} D_{i,2}^{(2)} A_{i,2}^{(2)} D_{i,2}^{(3)} A_{i,2}^{(3)} D_{i,2}^{(4)} A_{i,2}^{(4)} D_{i,2}^{(5)} A_{i,2}^{(5)} \cdots.$$

An attacker selects portions for $D_{i,1}^{(i)}$ and $D_{i,2}^{(i-1)}$, and then aligns two portions using 'alignment'. Experimental circumstance and setup are as follows:

Environment	PIC 16F84A microcontroller
Language	PIC programmer(Assembler)
Module	Scalar multiplication + sABS recoding + affine coordinate (Clock cycle of ECDBL: 3368)

First of all, for getting two distributions that is needed for judgement, we measure the three following power consumptions about time variable j.

- $S_1^{(i)}(j)$ related with the input point P_i.
- $S_2^{(i)}(j)$ related with the same input point P_i as before.
- $S_3^{(i)}(j)$ related with the different input point Q_i as before.

If the cryptographic device computes 160-bit scalar multiplications, $S_1^{(i)}(j)$ and $S_2^{(i)}(j)$ have 159 ECDBL signals for same point. Also, $S_1^{(i)}(j)$ and $S_3^{(i)}(j)$ have 159 ECDBL signals for different points. Before we define two distributions, the discriminant that can decide whether we have measured waveforms about the same operation or not is defined by

$$Disc.(S_1, S_2, t) = \frac{1}{m} \sum_{j=t+1}^{t+m} (S_1(j) - S_2(j))^2. \tag{4}$$

m is selected value in $[\lambda, n]$ where λ is the value including all coordinates x and y for the first time in ECDBL operation using an affine coordinate system and n is the length of signal related with 1 ECDBL.

Using this discriminant, we refer to two distributions X_1, X_2 as

$$X_1 = \bigcup_{i=1}^{L} \{Disc.(S_1^{(i)}, S_2^{(i)}, a) | a = k \times range, k \in \{0, 1, 2, ..., 158\}\},$$

$$X_2 = \bigcup_{i=1}^{L} \{Disc.(S_1^{(i)}, S_3^{(i)}, a) | a = k \times range, k \in \{0, 1, 2, ..., 158\}\}$$

$$where\ range\ is\ the\ length\ of\ signal\ related\ with\ 1\ ECDBL + 1\ ECADD$$
$$(range \approx \frac{approximate\ starting\ point\ of\ 159 - th\ ECDBL}{158}).$$

L is the value which decides the number of elements included in X_1, X_2.

In our research, sample rate is $100MS/s$, and so λ is about 60000 and n is about 336800 in equation (4). We select the value of m as 130000 in [60000, 336800]. Also, we select L as 3 in equation (5). Then, distributions X_1, X_2 are like a left side of Fig. 1. ($m_1 = E(X_1) = 24$, $a_1 = 63$, $m_2 = E(X_2) = 85$, $b_1 = 40$) where $E(\cdot)$ denotes the average of the distribution $\cdot$, a_1 denotes the maximum value of the distribution X_1, and b_1 denotes the minimum value of the distribution X_2.

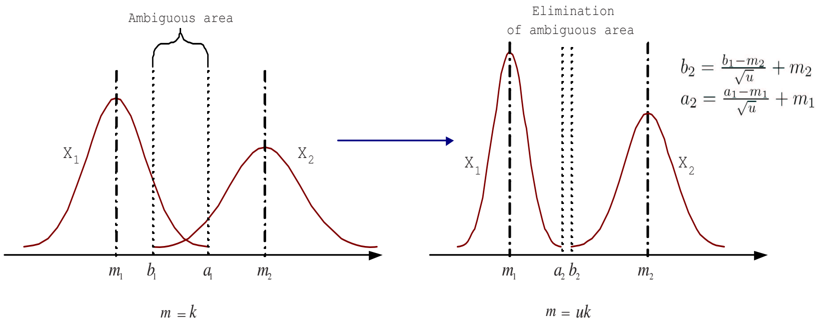

Ambiguous area
Elimination of ambiguous area
$b_2 = \frac{b_1 - m_2}{\sqrt{u}} + m_2$
$a_2 = \frac{a_1 - m_1}{\sqrt{u}} + m_1$
X1
X2
m_1
b_1
a_1
m_2
a_2
b_2
$m = k$
$m = uk$

Fig. 1. Distributions of ambiguous area and eliminated ambiguous area

An ambiguous area means the range that an attacker cannot decide whether signals related with the same operation or not. If the value of $Disc.(S_1, S_2, t)$ is in the ambiguous area ($40 \leq Disc.(S_1, S_2, t) \leq 63$ in ours), this value m for distinction must be selected in $[\lambda, n]$ to be the bigger value than the former. For reducing this error for each trial, the attacker must know the value of m that eliminates the ambiguous aria. In our experiment, we use the following proposition for eliminating this ambiguous area.

Proposition 1. *If $X_1 \leq a_1$, $X_2 \geq b_1$ are always completed with an error tolerance of $(\alpha/2)$ where $m = k$, the ambiguous area is eliminated where $m = (\frac{a_1 - b_1}{m_2 - m_1} + 1)^2 k$ with an error tolerance of $(\alpha - \alpha^2/4)$.*

Proof. According to the supposition, $P(X_1 \leq a_1) = P(X_2 \geq b_1) = \alpha/2$ where $m = k$.

If we convert this distributions into the standard normal distribution Z, the above equations are

$$P(Z \leq \frac{a_1 - m_1}{\sigma_1/\sqrt{k}}) = P(Z \geq \frac{b_1 - m_2}{\sigma_2/\sqrt{k}}) = \alpha/2 \ (\sigma_1^2 = var(X_1),\ \sigma_2^2 = var(X_2)).$$

If a_2, b_2 satisfy $P(X_1 \leq a_2) = P(X_2 \geq b_2) = \alpha/2$ where $m = uk$ (See the right side of Fig. 1.),

$$P(Z \leq \frac{a_2 - m_1}{\sigma_1/\sqrt{uk}}) = P(Z \geq \frac{b_2 - m_2}{\sigma_2/\sqrt{uk}}) = \alpha/2.$$

According to the above equations,

$$a_2 = \frac{a_1 - m_1}{\sqrt{u}} + m_1, \quad b_2 = \frac{b_1 - m_2}{\sqrt{u}} + m_2.$$

For eliminating the ambiguous area, the equation $b_2 > a_2$ must be satisfied, i.e.

$$u > (\frac{a_1 - b_1}{m_2 - m_1} + 1)^2.$$

The error tolerance (ET) is also $P(X_1 > a_2 \ or \ X_2 < b_2) = 1 - (1 - \alpha/2)^2 = (\alpha - \alpha^2/4)$. □

In the proposition, the value of α means the probability that the value of $Disc.(D_{i,1}^{(i)}, D_{i,2}^{(i-1)}, 0)$, in the practical attack, escapes previously measured bounds. This value depends on how many experiments have been carried out previously. If we compute the maximum ET when the ambiguous area is eliminated, these values are as follows according to the frequency L of the experiment.

L	1	2	3	4	5
ET	3.142×10^{-3}	1.572×10^{-3}	1.048×10^{-3}	7.860×10^{-4}	6.288×10^{-4}

In our preliminary research, distributions X_1 and X_2 about *Disc.*s can have 477 values for each, and so $P(X_1 > a_1) = P(X_2 < b_1)$ is less than 1/477 approximately. For this reason, because α is less than 1/954, the ET is also less than 1.048×10^{-3}. By proposition, we used m as 246514 and the discriminating value of *Disc.* as 52.321 ($= a_2 = b_2$). In other words, if the *Disc.* value of compared two ECDBL signals ($D_{i,1}^{(i)}$ and $D_{i,2}^{(i-1)}$ in recursive attack) is greater than 52.321, ECDBLs about different points have been carried out; otherwise, ECDBLs about same point have been carried out. Using these selected values through the preliminary research, we practically find the secret key comparing two ECDBL signals.

5 Countermeasures Against Proposed Attacks

In this section, we consider an environment that our attacks are applicable and a countermeasure to resist our attacks. First of all, our attacks and original DA can only be carried out in the affine coordinate. Suppose that the smart card uses a projective coordinate system. And then, even if ECDBL operations about the same point on an elliptic curve are carried out, values of each coordinate may not be different like Fig. 2. Hence, because values of coordinate are different, ECDBL signals for the same point could be considered as ECDBL about different points from the viewpoint of the attacker.

And now, we consider a countermeasure on proposed attacks. Because those use the method that chooses input points corresponding to the guessed bit of a recoded secret key, those can break recoding methods of the secret key for

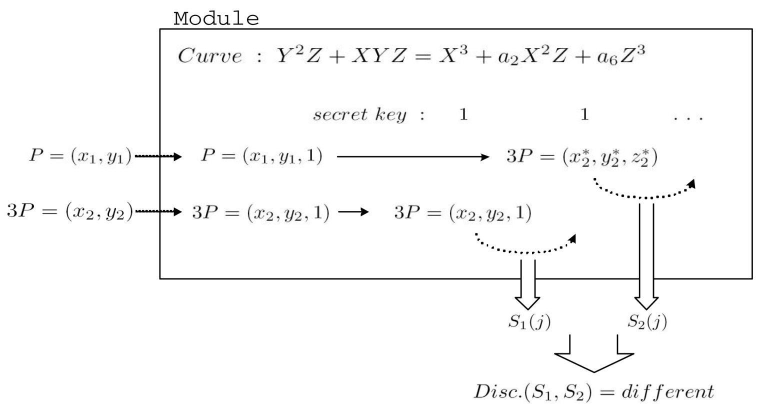

Module
$Curve : Y^2Z + XYZ = X^3 + a_2X^2Z + a_6Z^3$
$secret\ key :$ 1 1 . . .
$P = (x_1, y_1)$
$P = (x_1, y_1, 1)$
$3P = (x_2^*, y_2^*, z_2^*)$
$3P = (x_2, y_2)$
$3P = (x_2, y_2, 1)$
$3P = (x_2, y_2, 1)$
$S_1(j)$
$S_2(j)$
$Disc.(S_1, S_2) = different$

Fig. 2. The attack in projective coordinate system

SPA countermeasures. Hence, the countermeasure against our attacks must use random point. Various countermeasures using random point are introduced so far. Among them, BRIP [13] proposed by Mamiya et al. can be applied to our attacks efficiently.

6 Conclusion

In this paper we have proposed two attacks, recursive attack and initializing attack, against sABS countermeasure proposed by Hedabou et al. As these analyses classified into SPA the method that extend the DA, those enlarge the range of attack. We have performed an experiment to justify the possibility of our attacks. The concrete method of this experiment and the backing of the proposition can furnish an practical information about these analysis methods.

Acknowledgements

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement)" (IITA-2006-(C1090-0603-0025)). We would like to thank the anonymous referees for various comments and improvements.

References

1. ANSI X9.62, Public Key Cryptography for the Financial Services Industry, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 1999.
2. E. Biham and A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, CRYPTO 1997, LNCS 1294, pp. 513-525, 1997.

3. C. Clavier, M. Joye, *Universal Exponentiation Algorithm . A First Step towards Provable SPA-Resistance*, CHES 2001, LNCS 2162, pp. 300-308, 2001.
4. J.S. Coron, *Resistance against differential power analysis for Elliptic Curve Cryptosystems*, CHES 1999, LNCS 1717, pp. 292-302, 1999.
5. P.A. Fouque and F. Valette, *The Doubling Attack. Why Upwards Is Better than Downwards*, CHES 2003, LNCS 2779, pp. 269-280, 2003.
6. L. Goubin. *A refined power analysis attack on elliptic curve cryptosystems*, PKC 2003, LNCS 2567, pp. 199-211, 2003.
7. M.Hedabou, P.Pinel, and L. Bebeteau, *Countermeasures for Preventing Comb Method Against SCA Attacks*, ISPEC 2005, LNCS 3439, pp. 85-96, 2005.
8. ISO/IEC 15946-4, *Information technology - Security techniques . Cryptographic techniques based on elliptic curves - Part 4: Digital signatures giving message recovery.* Working Draft, JTC 1/SC 27, December 28th, 2001.
9. N. Koblitz, *Elliptic curve crypto- systems*, Math. of Computation, Vol.48, pp. 203-209, 1987.
10. P. Kocher, J. Jaffe, and B. Jun, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS,and Others Systems.* CRYPTO 1996, LNCS 1109, pp. 104-113, 1996.
11. P. Kocher, J. Jaffe, and B. Jun, *Introduction to differential power analysis and related attacks*, http://www.cryptography.com/dpa/technical, 1998.
12. P. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*, CRYPTO 1999, LNCS 1666, pp. 388-397, 1999.
13. H. Mamiya, A. Miyaji, H. Morimoto *Efficient Countermeasures Against RPA, DPA, and SPA*, CHES 2004, LNCS 3156, pp. 343-356, 2004.
14. Victor S. Miller, *Use of Elliptic Curves in Cryptography*, CRYPTO 1985, LNCS 218, pp. 417-426, 1985.
15. National Institute of Standards and Technology (NIST), *Recommended Elliptic Curves for Federal Government Use.* In the appendix of FIPS 186-2, available from http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf
16. R.L. Rivest, A. Shamir, and L.M. Adleman. *A method for obtaining digital signatures and public-key cryptosystem.* Communications of the ACM, 21(2):120-126, 1978.
17. Standards for Efficient Cryptography Group (SECG), *Specification of Standards for Efficient Cryptography*, Ver. 1.0, 2000. Available from http://www.secg.org/secg docs.htm
18. C.C. Tiu, *A New Frequency-Based Side Channel Attack forEmbedded Systems*, master's thesis, University of Waterloo, 2005.
19. Wireless Application Protocol (WAP) Forum, *Wireless Transport Layer Security (WTLS) Specification.* Available from http://www.wapforum.org
20. S.M. Yen, S.J. Kim, S.G. Lim, and S. J. Moon, *A countermeasure against one physical cryptanalysis May Benefit Another Attack*, ICISC 2001, Korea. Dec. 2001.

Securing the Distribution and Storage of Secrets with Trusted Platform Modules*

Paul E. Sevinç[1], Mario Strasser[2], and David Basin[1]

[1] Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
{paul.sevinc, basin}@inf.ethz.ch
[2] Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland
strasser@tik.ee.ethz.ch

Abstract. We present a protocol that allows servers to securely distribute secrets to trusted platforms. The protocol maintains the confidentiality of secrets in the face of eavesdroppers and careless users. Given an ideal (tamper-proof) trusted platform, the protocol can even withstand attacks by dishonest users. As an example of its use, we present an application to secure document processing.

1 Introduction

Trusted computing is about embedding a trusted computing base (TCB) [1] in a computing platform that allows a third party to determine the trustworthiness of the platform, i.e., whether or not the platform is a *trusted platform* from the point of view of a third party. The Trusted Computing Group (TCG), an industry standards organization, has specified a TCB for trusted computing in the form of three so-called roots of trust [2]: the *root of trust for storage* (RTS), the *root of trust for reporting* (RTR), and the *root of trust for measurement* (RTM). In particular, the TCG has specified a Trusted Platform Module (TPM) [3] that can act as both roots of trust for storage and measurement.[1] These specifications are clearly gaining momentum as witnessed by large-scale R&D projects such as *EMSCB* and *OpenTC*[2], open-source projects such as *TPM Emulator* [6] and *TrouSerS*[3], the inclusion of TPM services in *Windows Vista* [7], and the increasing number of personal computers with a TPM and a basic input/output system (BIOS) that can act as the RTM [8]. In the remainder of this paper, we understand "trusted computing" to mean trusted computing as specified by the TCG and focus on personal computers without limiting the paper's generality.

* This work was partially supported by the Zurich Information Security Center. It represents the views of the authors.

[1] There seems to be consensus in the information-security community that TCBs for trusted computing must be hardware-based, but tamper-proof hardware remains an open challenge [4,5].

[2] http://www.emscb.de/ and http://www.opentc.net/.

[3] http://tpm-emulator.berlios.de/ and http://trousers.sourceforge.net/.

Contribution. Our contribution in this paper is a protocol for securely distributing and storing secrets with TPMs. We specify the protocol in detail at the level of TPM commands and we informally analyze its security. The protocol is general in the sense that it is independent of a specific usage-control application. To illustrate how the protocol can be directly applied to nontrivial problems in usage control, we describe an application from the domain of secure document processing. In our specification, we treat the TPM as a trusted third party that can serve as an oracle for making platform measurements; this not only results in a clear specification, but this analogy for ideal roots of trust could also serve as the basis for a formal model of trusted computing in the future.

Organization. In Section 2, we provide a summary of those aspects of trusted computing that are of relevance to this paper. In Section 3, we discuss related work. In Section 4, we describe the problem that we solve with our protocol. We define the protocol and analyze its security in Sections 5 and 6, respectively. In Section 7, we draw conclusions. A concrete, realistic application scenario is presented in Appendix A.

2 Background

In this section, we summarize the TCG's definitions for root of trust for measurement, reporting, and storage. For a comprehensive description, the reader is referred to the TCG architecture overview [2] and to textbooks on trusted computing [9,10,11].

2.1 Root of Trust for Measurement

When a computer is booted, control passes between different subsystems. First the BIOS is given control of the computer, followed by the boot loader, the operating system loader, and finally the operating system. In an *authenticated boot*, the BIOS measures (i.e., cryptographically hashes) the boot loader prior to handing over control. The boot loader measures the operating system loader, and the operating system loader measures the operating system. These measurements reflect what software stack is in control of the computer at the end of the boot sequence; in other words, they reflect the platform configuration. Hence the name *platform configuration register* (PCR) for the TPM registers where such measurements are stored and which are initialized at startup and extended at every step of the boot sequence.

An attacker who wants to change the platform configuration without being detected has to corrupt the root of trust for measurement (in the BIOS), which we assume to be infeasible without physical access to the computer. Ideally, a tamper-proof piece of hardware will eventually act as the root of trust for measurement and measure the BIOS at the beginning of the boot sequence.

2.2 Root of Trust for Reporting

Each TPM has an *endorsement key* (EK) which is a signing key whose public key is certified by a trusted third party, such as the TPM manufacturer. For privacy reasons, the EK is only used to obtain a key certificate from a certificate authority (CA) for an *attestation identity key* (AIK), which the TPM generates itself. In order to alleviate even the strongest privacy concerns, direct anonymous attestation (DAA) [12,13] is the protocol of choice for certifying AIKs. AIKs are signing keys whose private key is only used for signing data that has originated in the TPM. For example, a remote party interested in learning what software stack is in control of the computer can query the TPM for PCR values. The query contains the set of PCRs to look up and a nonce (in order for the remote party to check for replay attacks). The TPM answers with the respective PCR values and the signature generated by signing the values as well as the nonce with one of its AIKs. Put differently, the TPM *attests* to, or *reports* on, the platform configuration.

2.3 Root of Trust for Storage

The protected storage feature of a TPM allows for the secure storage of sensitive objects such as TPM keys and confidential data. However, storage and cost constraints require that only the necessary (i.e., currently used) objects can reside inside a TPM; the remaining objects must be stored outside in unprotected memory and are revealed to the user or loaded into the TPM on demand. To this end, externally stored objects are encrypted (or *wrapped* in TCG terminology) with an asymmetric *storage key*, which is referred to as the parent key of the object. A parent key can again be stored outside the TPM and (possibly along with other keys) protected by another storage key. The thereby induced storage tree is rooted at the so called *storage root key* (SRK), which is created upon initialization of the TPM and cannot be unloaded. Consequently, a parent key has to be loaded into the TPM before the data it protects can be revealed or a key decrypted (or *unwrapped* in TCG terminology) and loaded into the TPM. Note that protected keys are only used inside the TPM and thus (in contrast to arbitrary data) are never disclosed to the user. Furthermore, each key is either marked as being *migrateable* or *non-migrateable*. In the former case, the key might be replicated and moved to other platforms whereas in the latter case the key is bound to an individual TPM and is never duplicated. Regarding the actual protection of objects, one differentiates between *binding* and *sealing*.

Binding is the operation of encrypting an object with the public key of a *binding key*. Binding keys are encryption keys. If the binding key is non-migratable, only the TPM that created the key can use its private key; hence, the encrypted object is effectively bound to a particular TPM.

Sealing takes binding one step further: the object is not only bound to a particular TPM, but in addition can only be decrypted if the current platform configuration matches the values associated with the protected object at the time of encryption.

It must be assumed that an attacker with physical access to the computer can get access to the private keys stored in the TPM. Current TPMs are designed to protect against software attacks, but not against hardware attacks (they are tamper-resistant at best) [10,11].

3 Related Work

Conceptually, properly measuring the software stack of a computer when the computer is booted (1), remotely attesting to a measurement and securely distributing secrets (2), and performing usage control on a computer once it has been deemed trustworthy and entrusted with the necessary secrets (3) are three straightforward tasks. As is often the case, the real complexity lies in the details.

Unlike the first task (e.g., [14,15,16]) and the third task (e.g., [17,18,19,20]), the second task has, until now, not been addressed in detail. Although protocols have been developed, those published are in the form of programming language-dependent and TPM library-dependent source code, without any security analysis. This may be the case because it is a deceptively simple task, which bears similarity with SSL/TLS. However, while the basic principles of SSL/TLS are fairly easy to understand, its details are quite intricate and the situation is similar here.

So far, the protocols for achieving the second task have been sketched at a very high level (similar to our summary in Figure 5 in Appendix A); for example in the form of the integrity-reporting protocol given in the TCG architecture overview [2, p. 9] and in the form of the two approaches for enhancing the protection of data on remote computers given by Pearson *et al.* [9, pp. 47-48]. This specification gap is filled in this paper.

4 Requirements

Consider the setting depicted in Figure 1: A server has secret data d_s that it is willing to share with certain clients over an open channel (i.e., one that is not encrypted in any way) upon request, but not with the clients' users. In practice (cf. the example given in Appendix A), the secret d_s may be a symmetric key K_D or the private key K_D^{-1} of an asymmetric key pair (K_D, K_D^{-1}). The owner of the secret d_s does not trust the users because they may not understand or respect the owner's security requirements or because they may have an untrustworthy platform, such as one compromised by a Trojan horse. In any case, the server is willing to share the secret d_s with clients who are known to meet the owner's security requirements. Because the main security goal of our protocol is *confidentiality* of the secret d_s, this entails that the client uses the secret d_s without disclosing it to the user or to any other entity. Furthermore, the client must either hinder the user from launching another process or at least force a change in the PCRs. Otherwise, the other (potentially malicious) process could simply request the TPM to disclose the secret d_s. It is in the server's interest to ensure that this security goal is met. Thus, the protocol needs to be resilient

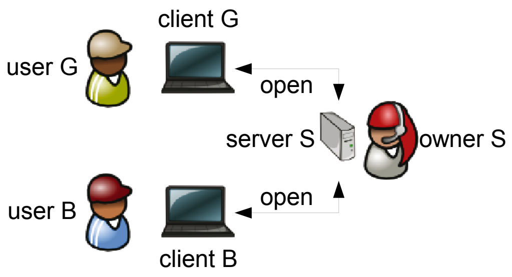


Fig. 1. Setting

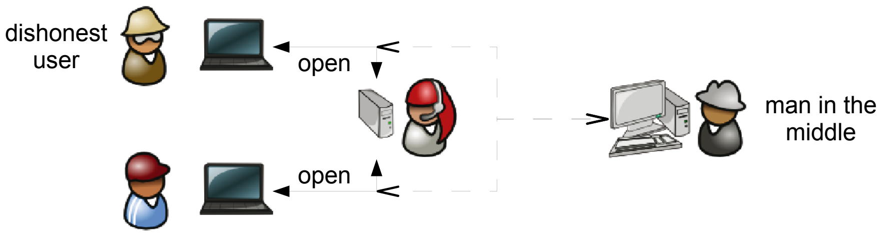


Fig. 2. Attackers

against man-in-the-middle attacks and, given an ideal (tamper-proof) trusted platform, against dishonest users as well (cf. Figure 2).

5 Protocol

In this section, we present a protocol that ensures that the server only distributes given secret data d_s to trusted clients (i.e., clients that meet the data owner's security requirements, as explained in the last section). The protocol involves three parties: the server, a client, and the client's TPM. Considering the TPM to be a participant in its own right may come as a surprise, but the following model should clarify this point.

Ideal roots of trust can be modeled as trusted third parties (cf. Figure 3) with certain oracle properties related to measurement. In particular, for each client there is a third party whom both the server and the client trust. Furthermore, the channel between the client and the trusted third party is secure (i.e., confidential and authentic). Not even the user can intercept or insert messages on this channel. There is no (direct) channel between the server and the trusted third party, though. Nevertheless, the server can encrypt data with the trusted third party's public encryption key along with information about a platform configuration. The trusted third party has the ability to determine the platform configuration of the client and decrypts data for the client only if the client's platform configuration is the one given when the data was encrypted.

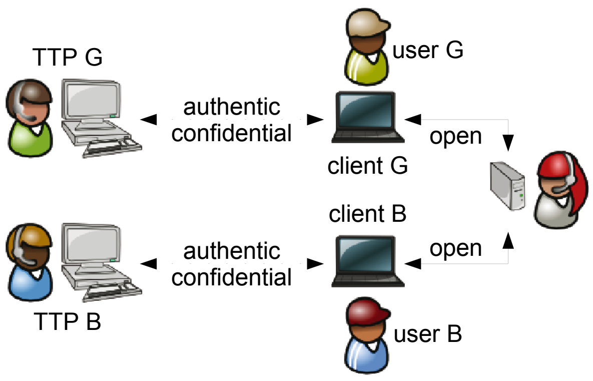


Fig. 3. Trusted Third Parties (TTPs) as a Model for Ideal Roots of Trust

In this model, our protocol basically proceeds as follows:

1. The client requests the secret d_s from the server.
2. The server encrypts d_s with the trusted third party's public key along with information about the platform configuration it trusts and sends the encrypted data to the client.
3. The client sends the encrypted data to the trusted third party and requests the trusted third party to decrypt the data for it.
4. The trusted third party determines the client's current platform configuration and reveals the decrypted data to the client only if the client's platform configuration is the one given when the data was encrypted.

Note that the first two steps only have to be taken once whereas the last two steps may be taken repeatedly.

The real protocol is more complex. In particular, it involves the generation of a platform configuration-dependent binding key and the use of an AIK. Even though privacy is not an issue for the kinds of application we originally had in mind, employing an AIK has the pleasant side effect of extending our protocol's usefulness to settings where privacy matters. For example, the secret data d_s could be the license key for a media player that manages digital rights. By using different AIKs when requesting license keys, the requests cannot be linked to the same client.

5.1 TPM Commands

We briefly introduce the TPM commands required in our protocol. For the sake of simplicity, we omit input values such as command-authorization data and key parameters as well as output values such as error codes. For a comprehensive description of the commands, the reader is referred to the TPM main specification [3] and to Pearson *et al.* [9].

TPM_CreateWrapKey generates an asymmetric key and returns the public key in plain text and the private key encrypted with the key pair's parent key. The input values of interest to us are a key handle that points to the generated key's parent key, the flag which declares the key as a binding or signing key, the flag which declares the key as migratable or non-migratable, and the (potentially empty) set of PCRs to whose values the key is sealed. Note that for the key to be non-migratable, the parent key must be non-migratable as well.

In our protocol, we use this command on the client to generate a non-migratable binding key that is sealed to a (non-empty) set of PCRs.

TPM_LoadKey2 loads an asymmetric key onto the TPM and returns the key handle that points to the loaded key, thus making the key available for use in subsequent TPM commands. The input values of interest to us are the public key, the encrypted private key, and a key handle that points to the loaded key's parent key. A non-migratable key will only be loaded onto the TPM if it was generated by the TPM.

In our protocol, we use this command on the client to load the binding key generated with TPM_CreateWrapKey onto the client TPM.

TPM_CertifyKey returns a key certificate. The input values of interest to us are a key handle that points to the key to certify and a key handle that points to the certifying signing key.

In our protocol, we use this command on the client to certify with an AIK that the binding key generated with TPM_CreateWrapKey and loaded with TPM_LoadKey2 is a non-migratable binding key that is sealed to a set of PCRs.

TSS_Bind encrypts data and returns it in cipher text. The input values of interest to us are the data to encrypt and the public key used for encryption. Note that it is the responsibility of the caller to ensure that the encryption key is a non-migratable binding key. Note further that the TSS_Bind command is not a TPM command, but fully implemented in software.

In our protocol, we use this command on the server to encrypt a secret with the public key of the binding key generated with TPM_CreateWrapKey and certified with TPM_CertifyKey.

TPM_UnBind decrypts data and returns it in plain text. The input values of interest to us are the data to decrypt and a key handle that points to the binding key (whose private key is used for decryption). A sealed binding key will only be used by the TPM if the values in the PCRs match those specified during sealing.

In our protocol, we use this command on the client to decrypt the secret encrypted with TSS_Bind with the private key of the binding key generated with TPM_CreateWrapKey and loaded with TPM_LoadKey2.

5.2 Notation

We employ so-called "Alice & Bob" notation, which leaves implicit many of the checks carried out by principals when executing the protocol and the associated control flow when these checks fail, e.g., aborting when the verification of a digital signature fails [21]. Nevertheless, we have annotated our protocol specification, making explicit the checks of TPM-specific key properties and platform configuration information, using an assert statement. The semantics of assert is standard: it aborts the protocol execution when the asserted predicate does not hold. Furthermore, we employ the following notation:

– REQ is a constant, requesting the secret data d_s.
– PCR_INFO is a set of PCR indices and their respective values.
– $\mathcal{K}_X$ is a key pair with public key K_X and private key K_X^{-1}.
– H_X is the handle to the key $\mathcal{K}_X$.
– *aik*, *binding*, and *non-migratable* are flags that denote properties of a key, namely that the key is an AIK, a binding key, and non-migratable, respectively.
– $Enc_P(\mathit{binding}, \mathit{non\text{-}migratable}, PCR\_INFO, K_C^{-1})$ is the private key K_C^{-1} of the non-migratable binding key $\mathcal{K}_C$ sealed to PCR_INFO, encrypted with the (non-migratable) parent key $\mathcal{K}_P$.
– $Sig_{AIK}(\mathit{binding}, \mathit{non\text{-}migratable}, PCR\_INFO, N, K_C)$ is the certificate of the public key K_C of the non-migratable binding key $\mathcal{K}_C$ sealed to PCR_INFO, signed with the private key K_{AIK}^{-1} of the signing key $\mathcal{K}_{AIK}$.
– $Sig_{CA}(aik, K_{AIK})$ is the certificate of the public key K_{AIK} of the attestation identity key $\mathcal{K}_{AIK}$, signed with the private key K_{CA}^{-1} of the signing key $\mathcal{K}_{CA}$.

5.3 Initial Possessions of Parties

The **server** knows

– the secret data d_s,
– the public key K_{CA} of the certificate authority's signing key, and
– the PCRs and their values for the trusted stack.

The **client** knows

– the handle H_P to (and authorization data for) a non-migratable storage key $\mathcal{K}_P$,
– the handle H_{AI} to (and authorization data for) an AIK, and
– the certificate $Sig_{CA}(aik, K_{AIK})$ for verification of the AIK by a third party, in our case the server.

In a protocol run, the AIK allows the client to prove to the server that the latter is indirectly interacting with a TPM.
The **TPM** knows (i.e., has loaded)

– the private key K_P^{-1} of the non-migratable storage key $\mathcal{K}_P$ and
– the private key K_{AI}^{-1} of the AIK.

Table 1. Key Distribution Protocol

1	C $\longrightarrow$ S	REQ
2	C $\longleftarrow$ S	PCR_INFO, N
3	TPM $\longleftarrow$ C	`TPM_CreateWrapKey`$(H_P, \textit{binding}, \textit{non-migratable}, PCR\_INFO)$
4	TPM	assert $\mathcal{K}_P$ is *non-migratable*
		generate non-migratable binding key (K_C, K_C^{-1})
5	TPM $\longrightarrow$ C	$K_C, Enc_P(\textit{binding}, \textit{non-migratable}, PCR\_INFO, K_C^{-1})$
6	TPM $\longleftarrow$ C	`TPM_LoadKey2`$(K_C,$ $Enc_P(\textit{binding}, \textit{non-migratable}, PCR\_INFO, K_C^{-1}), H_P)$
7	TPM $\longrightarrow$ C	H_C
8	TPM $\longleftarrow$ C	`TPM_CertifyKey`(H_C, H_{AIK}, N)
9	TPM $\longrightarrow$ C	$Sig_{AIK}(\textit{binding}, \textit{non-migratable}, PCR\_INFO, N, K_C)$
10	C $\longrightarrow$ S	$Sig_{AIK}(\textit{binding}, \textit{non-migratable}, PCR\_INFO, N, K_C),$ $Sig_{CA}(aik, K_{AIK})$
11	S	assert K_{AIK} is *aik*
		assert K_C is *binding*
		assert K_C is sealed to PCR_INFO
12	C $\longleftarrow$ S	$Enc_C(d_s)$
13	TPM $\longleftarrow$ C	`TPM_UnBind`$(Enc_C(d_s),\ H_C)$
14	TPM	assert C is in state PCR_INFO
15	TPM $\longrightarrow$ C	d_s

Protocol Run. The protocol is specified in Table 1. The client initiates a protocol run by requesting the secret data d_s from the server (1). The server replies with the set of PCRs that have to be used to represent the (trusted) state of the client and a nonce N which identifies the protocol run (2). Note that the nonce does not provide additional security since replay attacks are not an issue.[4] The client invokes the `TPM_CreateWrapKey` command (3) to have the TPM create a non-migratable asymmetric encryption key $\mathcal{K}_C := (K_C, K_C^{-1})$ (4) that is sealed to the PCRs specified in step 2 (5). The client loads the key into the TPM by invoking `TPM_LoadKey2` (6), receives the key handle from the TPM (7), and has the TPM certify the loaded key with the AIK (8). The TPM returns the certificate (9), which the client forwards to the server together with the certificate of the AIK (10). The server checks that the certificates are valid (11), in particular that $\mathcal{K}_C$ is a non-migratable binding key sealed to the required

[4] The reason is that it is not important when the binding key has been generated and certified, but what its properties are. Because the binding key is non-migratable, these properties (in particular, which TPM it is associated with) never change. So even though the `TPM_CertifyKey` command is specified to take a nonce as argument, the nonce could be replaced with something predictable (and more efficiently implemented) like a counter in our protocol.

PCRs. If the key $\mathcal{K}_C$ has the required properties, the secret d_s is bound to it and the resulting protected object returned to the client (12). Upon receipt of the protected object, the client invokes `TPM_UnBind` to have the TPM decrypt the secret d_s (13). The TPM checks that the client is in the trusted state (14) before using $\mathcal{K}_C$ and returning the secret d_s (14).

Note that there is no need for (and no additional security in) explicitly sealing the secret d_s since the private key K_C^{-1} of the binding key $\mathcal{K}_C$ is sealed to the required PCRs itself. From now on, whenever the client is in the trusted state (i.e., the PCR values match the required ones) it can have the TPM unbind the secret d_s for its intended use.

6 Security Analysis

6.1 Security Against Man-in-the-Middle Attacks

Since the communication channel between the client and the server is open, a man in the middle sees the following four messages exchanged in steps 1, 2, 10, and 12:

1. REQ,
2. $PCR\_INFO,\ N$,
3. $Sig_{AIK}(\textit{binding}, \textit{non-migratable}, PCR\_INFO, N, K_C)$,
   $Sig_{CA}(aik, K_{AIK})$, and
4. $Enc_C(d_s)$

Obviously, the first three messages are independent of the secret data d_s, and hence they provide no information about it, even in a strict information-theoretic sense. The fourth message is encrypted with the public key K_C, where the corresponding private key K_C^{-1} is unknown to the man in the middle. Hence, the man in the middle can only decrypt the fourth message by breaking the cryptographic system, which we assume to be infeasible, or by deriving the private key K_C^{-1} from the first three messages. However, the first two messages are also completely independent of the encryption key $\mathcal{K}_C$ and deriving a private key from the corresponding public key in the third message amounts to breaking the cryptographic system. Thus, the protocol is also secure against a passive man-in-the-middle attack.

An active man in the middle could try to replace the third message he sees by $Sig_{MS}[enc,\ nm,\ PCR\_INFO,\ N](K_{ME}), Sig_{CA}[sig](K_{MS})$—$\mathcal{K}_{MS}$ is his signing key and $\mathcal{K}_{ME}$ is his encryption key—in order to fool the server into binding the secret d_s to $\mathcal{K}_{ME}$ ($Enc_{ME}(d_s)$). However, this requires forging the CA's signature, which again amounts to breaking the cryptographic system. Thus, the protocol is also secure against an active man-in-the-middle attack.

6.2 Ideal Trusted Platform: Security Against Dishonest Users

Because the server verifies the two certificates exchanged in a protocol run, in particular that $\mathcal{K}_C$ is non-migratable and sealed to the PCRs specified by PCR_INFO, and because it binds the secret data d_s to $\mathcal{K}_C$, the client must execute step 13 of the protocol as the honest client (i.e., using a trusted software stack). Afterwards, a dishonest user could put the client into a dishonest state (by launching another process if at all permitted or rebooting another stack) which results in different PCR values and in the TPM not unbinding the secret d_s. Alternatively, a dishonest user could mount a hardware attack in order to read the secret d_s out of the TPM, which we assume to be infeasible given an ideal (tamper-proof) trusted platform. Thus, an ideal (tamper-proof) trusted platform not only provides security against man-in-the-middle attacks but also against attacks by dishonest clients.

7 Conclusion

We have presented in detail a protocol at the level of TPM commands that allows servers to securely distribute secrets to trusted platforms. The protocol maintains the confidentiality of secrets in the face of eavesdroppers and careless users. Given an ideal (tamper-proof) trusted platform, the protocol maintains the confidentiality of secrets even in the face of dishonest users.

We have provided an informal analysis of the security of the protocol. A formal analysis of our protocol and other TPM-based protocols would require developing formal models for TPM-specific concepts such as binding and sealing. This is an interesting topic for future work that could be based on groundwork laid by Lin [22].

Acknowledgments

We would like to thank Michael Näf for his feedback on this paper and Thomas Zweifel for his previous collaboration on this topic. We also thank the artists who contribute to the Open Clip Art Library.

References

1. Bishop, M.: Computer Security: Art and Science. Addison Wesley Professional (2003)
2. Trusted Computing Group: TCG architecture overview. (TCG Specification)
3. Trusted Computing Group: TCG TPM specification version 1.2. (TCG Specification)
4. Anderson, R., Bond, M., Clulow, J., Skorobogatov, S.: Cryptographic processors – a survey. Technical Report 641, University of Cambridge (2005)
5. Smith, S.W.: Trusted Computing Platforms: Design and Applications. Springer-Verlag (2005)
6. Strasser, M.: A software-based TPM emulator for Linux. Semesterarbeit, ETH Zurich (2004)
7. Microsoft: Windows Vista beta 2 trusted platform module services step by step guide. (Published on the WWW)
8. Kay, R.L.: This ain't your father's internet: How hardware security will become nearly ubiquitous as a rock solid solution to safeguarding connected computing. (Published on the WWW)

9. Pearson, S., ed.: Trusted Computing Platforms: TCPA Technology in Context. Prentice Hall (2003)
10. Grawrock, D.: The Intel Safer Computing Initiative. Intel Press (2006)
11. Mitchell, C., ed.: Trusted Computing. Volume 6 of IEE Professional Applications of Computing. The Institution of Electrical Engineers (2005)
12. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In Pfitzmann, B., Liu, P., eds.: Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004), ACM Press (2004) 132–145
13. Camenisch, J.: Better privacy for trusted computing platforms. In Samarati, P., Ryan, P., Gollmann, D., Molva, R., eds.: Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS 2004). Volume 3193 of Lecture Notes in Computer Science., Springer-Verlag (2004) 73–88
14. Marchesini, J., Smith, S.W., Wild, O., MacDonald, R.: Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. Technical Report Dartmouth TR2003-476, Dartmouth College (2003)
15. Marchesini, J., Smith, S.W., Wild, O., Barsamian, A., Stabiner, J.: Open-source applications of TCPA hardware. In: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004), IEEE Computer Society (2004) 294–303
16. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. In: Proceedings of the 13th Usenix Security Symposium. (2004) 223–238
17. Sailer, R., Jaeger, T., Zhang, X., van Doorn, L.: Attestation-based policy enforcement for remote access. In: Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004). (2004) 308–307
18. Sandhu, R., Zhang, X.: Peer-to-peer access control architecture using trusted computing technology. In: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT 2005). (2005) 147–158
19. Zhang, X., Chen, S., Sandhu, R.: Enhancing data authenticity and integrity in p2p systems. IEEE Internet Computing **9**(6) (2005) 42 –49
20. Sandhu, R., Ranganathan, K., Zhang, X.: Secure information sharing enabled by trusted computing and pei models. In: Proceedings of the 2006 ACM Conference on Computer and Communications Security (ASIACCS 2006), ACM Press (2006) 2–12
21. Caleiro, C., Viganò, L., Basin, D.: Deconstructing Alice and Bob. In: Proceedings of the Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA 2005). Volume 135 of Electronic Notes in Theoretical Computer Science. (2005) 3–22
22. Lin, A.H.: Automated analysis of security APIs. Master's thesis, Massachusetts Institute of Technology (2005)
23. Sevinç, P.E., Basin, D., Olderog, E.R.: Controlling access to documents: A formal access control model. In Müller, G., ed.: Proceedings of the 1st International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006). Volume 3995 of Lecture Notes in Computer Science., Springer-Verlag (2006) 352–367
24. Sevinç, P.E., Basin, D.: Controlling access to documents: A formal access control model. Technical Report 517, ETH Zurich (2006)
25. Sevinç, P.E.: Securing Information by Controlling Access to Data in Documents. PhD thesis, ETH Zurich (2007)

A Application: Document Security

We apply our protocol in situations where access to documents of an enterprise must be controlled on computers that are not owned and administrated by the enterprise. For example, members of the enterprise's board of directors may be more inclined to read documents on a computer they already have, such as their home computer, than to be issued a computer from every enterprise on whose board of directors they sit.[5]

Such a situation is depicted in Figure 4. The main entities are a user, the user's computer (which the user owns and administrates), and a document whose content is confidential. Access to the document is governed by its access policy. The document content and the document policy are paired. The document's content is encrypted and the document as a whole signed such that the user can neither directly access the content component (because he does not know the decryption key) nor alter the policy in his favor (because he does not know the signing key). Instead, the user has to access the document via a document processor on an operating system that the document owner trusts to enforce the policy and to maintain confidentiality.

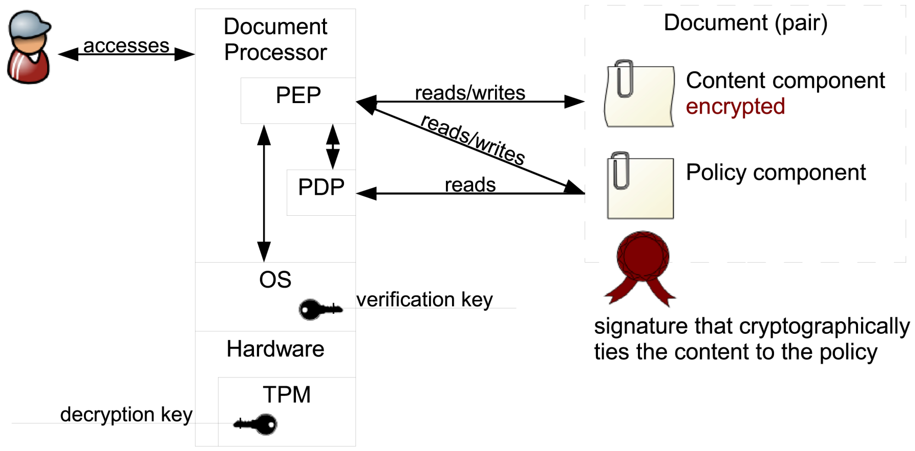


Fig. 4. Key Use

Within the document processor, the policy enforcement point (PEP) is responsible for enforcing the policy based on access decisions made by the policy decision point (PDP). Upon opening a document, the document processor has to

[5] Recall the case of former CIA Director John Deutch who accessed classified material on his unsecured home computer. The problem was not that he was not trusted (as the CIA director, he certainly was), but his software might have been untrustworthy. Had the classified material been encrypted with a key known not even to him, he could have been forced to boot his home computer into a trusted state, and the story would never have made the news.

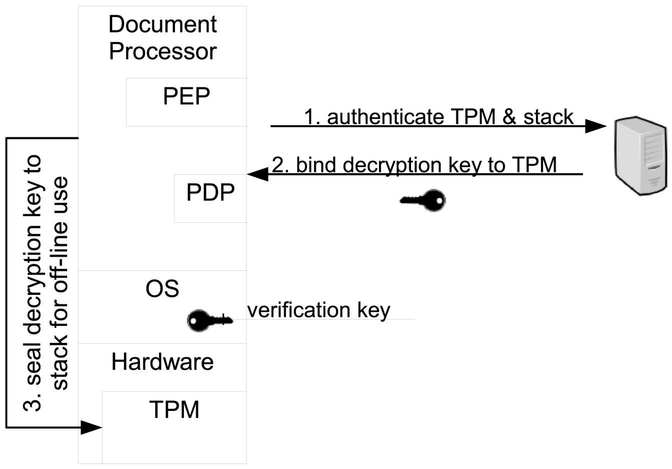

Document Processor
PEP
PDP
OS
Hardware
TPM
1. authenticate TPM & stack
2. bind decryption key to TPM
3. seal decryption key to stack for off-line use
verification key

Fig. 5. Key Distribution

verify the document's authenticity (which implies its integrity) and decrypt the content. The decryption key is sealed to the trusted software stack (document processor and operating system) such that it can be used off-line but not accessed with another software stack in control. Note that the user cannot replace the verification key without changing (the hash of) the stack.

We achieve this situation in three main steps (cf. Figure 5):

1. The document owner's server checks whether the trusted stack is in control of the user's computer (steps 1–11 of the key distribution protocol).
2. It binds the decryption key to the user's TPM (step 12).
3. The trusted stack seals the decryption key in the user's TPM (steps 13–15).

In previous work [23,24,25], we developed an access-control system for documents. Standard operating system security mechanisms can be used to ensure that the system is not tampered with within an enterprise. This work is the missing link to ensuring that the system cannot be circumvented outside of the enterprise.

Distributed Certified Information Access for Mobile Devices[⋆]

Aniello Del Sorbo[1], Clemente Galdi[2], and Giuseppe Persiano[1]

[1] Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università degli Studi di Salerno
Via Ponte Don Melillo - 84084 Fisciano (SA) - Italy
{anidel,giuper}@dia.unisa.it
[2] Dipartimento di Scienze Fisiche
Università degli Studi di Napoli "Federico II"
Complesso Univ. Monte S. Angelo - Via Cinthia
80126 Napoli - Italy
galdi@na.infn.it

Abstract. In this paper we describe a primitive, which we call, Certified Information Access, in which a database answers to a query by providing the information matching the query along with a proof that such information are consistent with the actual content of the database. We show that such a primitive can be securely implemented in a distributed fashion. Furthermore, we describe the design principles for a distributed architecture that would allow the use of this primitive on mobile devices.

Keywords: Secure protocols, Secure services for mobile devices.

1 Introduction

The growing need of mobility in the current society and the increasing availability of low-cost wireless devices have fostered an impressive growth in the number of services available for such devices. Currently available technologies allow the possibility of performing tasks on wireless devices that were impossible only few years ago. Theoretically, it is now possible to run any Java program on last generation mobile devices. On one hand, this allows the possibility of interaction between any Java-enabled mobile device with any possible application that supports web-based access. It is possible to download programs from the Internet and execute them in order to interact with a specific service.

This flexibility poses a number of security issues that need to be addressed. Just to mention a few, authentication, anonymity, accountability of users/services need be to thought for a environment in which the device has very low computational abilities, the communication medium can be easily eavesdropped or subject to malicious attacks of various kind, etc. In particular, because of the small computational power of such devices, one issue to address is the performance of the applications.

⋆ This work was partially supported by the European Union under IST FET Integrated Project AEOLUS (IST-015964).

In a such global scenario it is often the case that one entity has to access information owned by a different entity. This poses a number of security issues both from the database side and the user side. For example, the owner of the database may require that only authorized users have access to the information or part of it. This is an instance of the well-known access control problem. Another example could be that each user cannot infer additional knowledge by analyzing the answers to the issued queries. In other words, authorized users can obtain the information they required, but they cannot infer any other information from the received answers [5,9]. On the other hand, a user may require that the database does not gain any information about specific content she requested [6,4,10,11,1]. This means that the database should "blindly" answer all the queries from authorized users in a way that all the requested information can be correctly reconstructed from the answers.

The above problems mainly address the confidentiality of the information. The primitive we consider in this prototype, introduced in [8], deals with the problem of guaranteeing the consistency of the answer, sent by the database to the user, with the information actually contained in the database. We assume the possibility that a database would be willing to give wrong answer to queries issued by a user. In this setting, since the user does not know in advance which is the actual information he will receive from the database, there would be no way to distinguish between a correct answer from a maliciously modified one.

Certified Information Access (CIA for short) primitives force the database to publish a snapshot of its current contents, which we refer to as the *Public Information*, on a trusted entity. After such information is available, the user may issue queries to the database. The answers to each query have to be *consistent* with the public information.

One trivial way of implementing such primitives is to publish the whole content of the database on a trusted server. In this case, a user may compare the received query with the one contained in the public copy of the database. This solution is, of course, neither secure nor efficient. Since the database has to publish its whole contents, the confidentiality of the information therein contained is compromised. Furthermore, since all the elements in the database need to be transmitted and stored, the communication and space complexities of such solution is linear in the size of the database.

For this reasons the public information should satisfy the following properties:

– *Compactness:* The size of the public information should be smaller than the size of the database.
– *Confidentiality:* The public information should not reveal anything about the actual content of the database.
– *Correctness:* A correct answer to a query should be consistent with the public information with probability one.
– *Soundness:* Any wrong answer to a query will be detected with high probability.

Currently, a way of implementing CIA primitives is by means of a new cryptographic primitive, namely *mercurial commitments* introduced and studied in

[8,3,2,7]. Unfortunately, the implementation of such primitive are computationally intensive. On one hand, the generation of the public information is time consuming also on current servers. On the other hand, although the verification procedure can be easily executed on a PC in few seconds, it still requires much more time on mobile devices.

Our Contribution. In this paper we present a distributed architecture for a CIA service. We first describe in details the Certified Information Access service. We briefly review the basic primitives that can be used for implementing such a service. However, such primitives require a certain amount of computation that would make any solution for CIA unfeasible on mobile devices or, more generally, on devices with low computational power. We show that such primitives can be computed securely in a distributed fashion. In other words, any device can distribute its load securely among untrusted *peers* and locally combine the results of such computations. We finally show an architectural design for the distributed implementation of a secure CIA service.

We describe a solution for *static* databases, i.e., databases in which the content does not change. To the best of our knowledge, the *only* secure solution to the problem of dynamic databases is the one described in [7]. Unfortunately this solution is not efficient. Indeed, updating a single element in the database results in the need of storing some information whose size is linear in the size of the key.

We assume that there exists a trusted entity that does not collude with the entity holding the database. Furthermore, the system comprises a sufficient number of peers whose only role is to compute modular exponentiations. We assume that such peers are honest, that is, they properly execute the protocols, but a small fraction of them may be curious, in the sense that they may collude in order to infer additional information from the messages they have exchanged.

This document is organized as follows: In Section 2 we describe the CIA primitive. In Section 3 we describe a basic tool that is used in the prototype, namely Mercurial Commitment schemes, and techniques for distributing the computation of such primitive among the peers. In Section 4 we describe how to construct a certified information access primitives using mercurial commitments. In Section 5 we report the design principles for a distributed architecture implementing a CIA service.

2 Certified Information Access

In this section we describe the CIA functionality that we will be at the base of our prototype.

In the context of secure databases, an implementation of a certified information access has to provide the users with a database service in which each answer to a query consists of the actual query results and a proof that such information is indeed the actual content of the database. The verification of the proof can be accomplished by using some public information that the database provided before the query was issued. Such public information should not reveal anything about the actual content of the database. In a CIA system we identify three parties, the CertifiedDBOwner the User and the PubInfoStorage.

In a setup phase, the PubInfoStorage generates the public parameters that will be used for the CIA service. The PubInfoStorage is assumed not to collude with the CertifiedDBOwner.

The CertifiedDBOwner, based on public parameters and the content of the database, produces the public information that is then sent to the PubInfoStorage. Whenever a User makes a query to the CertifiedDBOwner, he obtains an object that contains the answer to the query and some information that can be used, along with the information held by the PubInfoStorage, to prove that the answer is indeed correct and that the CertifiedDBOwner has not cheated.

Cryptographic background. A very simple type of Certified Information Access, is the one in which the database consists of only one string, one-string CIA (or 1-CIA). The one-string CIA functionality has been studied in Cryptography under the name of *commitment* and several implementations of this primitive have been presented. Instead, the concept of a M1-CIA corresponds to a special type of commitments called *mercurial* commitments introduced by [3] and later studied by [2,7].

1-CIA can be seen as a safe. The CertifiedDBOwner writes the string m on a piece of paper, puts it in a safe and locks the safe. The safe can be sent to a User that cannot open it (thus guaranteeing the *hiding* property). On the other hand the User is guaranteed that the message in the safe cannot change while it is in the safe (thus guaranteeing the *binding* property). Whenever the CertifiedDBOwner chooses to, he can *open* the safe by sending the string m and the key to open the safe. The User can then *verify* that the value he sees is the same as the original message stored in the safe.

Using a commitment scheme, we can implement a 1-CIA as follows: the CertifiedDBOwner, based on the actual value m of the string, produces a *commitment* and a *decommitment key*. The commitment is sent to the PubInfoStorage. Whenever the CertifiedDBOwner chooses to, he *opens* the commitment by releasing the the value of m and the decommitment key. The User can verify that the opening has been correctly performed by checking the open against the information held by the PubInfoStorage.

The M1-CIA functionality is an extension of the 1-CIA functionality with an important extra property. In addition to the usual operation of opening a commitment, M1-CIA also supports a *partial open* operation called *tease*. A commitment `com` can be computed in two ways: it can be a *hard* commitment, that is a commitment that can be opened and teased in only one way; or a *soft* commitment that cannot be opened at all, but can be teased to any value.

The *binding* and *hiding* properties also hold for a M1-CIA. In addition, hard commitments are indistinguishable from soft ones. In particular, this means that it is computationally infeasible to distinguish whether a commitment `com` is a soft or a hard one.

The mechanism is the same as the ones of the 1-CIA. The only difference is that the CertifiedDBOwner can also produce soft commitment that can be teased to any string m.

A M1-CIA scheme provides the following functions.

CertifiedDBOwner

- Commit: computes, on input the string m and the public parameters, the *commitment* `com` to be sent to PubInfoStorage and the *decommitment key* `dec` to be used in the opening.
- SoftCommit: computes, on input the public parameters, a *soft commitment* `Scom` along with a *teasing key* `Sdec` to be used for teasing `Scom`.
  Notice that SoftCommit does not need a string m as `Scom` can be teased to any value m.
- Tease: computes, on input the public parameters, a string m, a commitment `com` (`com` could be a hard or a soft commitment) and a teasing key `Sdec`, the *teasing* τ of `Scom` to string m.

User

- VerifyOpen: Given the public parameters, verifies that message m and a decommitment key `dec`, are consistent with a commitment `com`.
- VerifyTease: verifies, on input the public parameters, that a teasing τ of a commitment `com` (it could be a soft commitment or a hard one) to a string m has been correctly computed.

3 Primitives

In this section we show how we implement the M1-CIA functionality. Our implementation is based on the hardness of the discrete logarithm in cyclic groups and is based on [3].

3.1 Implementing M1-CIA Via Mercurial Commitments

The Setup procedure (executed by the PubInfoStorage) consists in randomly picking a random prime p and two generators g, h of the cyclic group $Z_p^\star$. All operations are to be considered in the group $Z_p^\star$ unless otherwise specified.

The Commit procedure (executed by the CertifiedDBOwner) takes as input the string m and public parameters (p, g, h) and computes `com` and `dec` as follows: randomly pick $r_0, r_1 \in Z_{p-1}^\star$ and set `com` $= (g^m \cdot (h^{r_1})^{r_0}, h^{r_1})$ and `dec` $= (r_0, r_1)$.

The VerifyOpen procedure (executed by the User) takes as input the public parameters (p, g, h), a message m, a commitment `com` $= (C_0, C_1)$ and decommitment key `dec` $= (r_0, r_1)$ and consists in checking that $C_0 = g^m \cdot C_1^{r_0}$ and $C_1 = h^{r_1}$.

The SoftCommit procedure (executed by the CertifiedDBOwner) takes as input the public parameters (p, g, h) and computes $\texttt{Scom}$ and $\texttt{Sdec}$ as follows: randomly pick $r_0, r_1 \in Z^\star_{p-1}$ and set $\texttt{Scom} = (g^{r_0}, g^{r_1})$ and $\texttt{Sdec} = (r_0, r_1)$.

The teasing τ of a hard commitment $\texttt{com} = (g^m \cdot (h^{r_1})^{r_0}, h^{r_1})$ of string m with the decommitment key $\texttt{dec} = (r_0, r_1)$ consists simply of $\tau = r_0$.

Instead the teasing τ of a soft commitment $\texttt{Scom} = (g^{r_0}, g^{r_1})$ with teasing key $\texttt{Sdec} = (r_0, r_1)$ to string m is computed by setting $\tau = (r_0 - m)/r_1 \pmod{p-1}$.

The VerifyTease procedure (executed by the User) takes as input public parameters (p, g, h) and teasing τ of commitment (C_0, C_1) to string m consists in checking that $C_0 = g^m \cdot C_1^\tau$.

Correctness and security of this scheme have been shown in [3].

3.2 Distributing M1-CIA Computation

This paragraph describes a way of distributing the computations needed to create and verify mercurial commitments while preserving the security of the M1-CIA scheme.

Since the most time-consuming operation is the modular exponentiation, we show a way of distributing such operation securely. The idea behind the load distribution is to use the computational power of peers to execute modular exponentiations. In this way the CertifiedDBOwner and the User are only required to execute additions and multiplications.

Secure Computation of Exponentiations. Crucial operations to be distributed are modular exponentiations in which either the exponent or both the base and the exponent are sensitive information. We assume that a service Mod_Exp is run a set of peers that the User and the CertifiedDBOwner may use for such operations. Peers are assumed to be honest, i.e., compute correctly the modular exponentiations they are required to, but curious, in the sense that they may collect the information received in order to obtain information on the values queried by the user or on the elements of the database.

Mod_Exp*:* We assume that the peers can be identified by an ID in the set $\{1, \ldots, t\}$, for some integer t. This service is inkoved with input a base b, an exponent r, the modulus p and the ID of the peer that will execute the task. The peer with identity ID computes $b^r \bmod p$ and sends back the result to the player who required it. We assume secure point-to-point communication between peers and the player who invokes their services.

Computation with secure exponent. We first analyze the case in which the exponent is a secret information. Let k be an integer such that $k - 1 \ll t$. We assume that the maximum number of colluding peers is at most $k - 1$. In this case, given b and e, it is possible to compute $b^e \bmod p$ keeping the exponent e private as follows:

Procedure SECURE_EXP(b, e, p, k)

- randomly select k out of the t peers and let $\{ID_1, \ldots, ID_k\}$ be their identities.
- pick $e_1, \ldots, e_{k-1} \in_R Z_{p-1}$
- $e_k = e - (e_1 + \ldots + e_{k-1}) \bmod (p-1)$.
- $r_i = \text{MOD\_EXP}(b, e_i, p, ID_i)$, for $i = 1, \ldots, k$.
- $r = \prod_{i=1}^{k} r_i \bmod p$
- output r

The correctness of the above procedure follows immediately from the fact that $r_i = b^{e_i} \bmod p$ and, thus, $r = b^{e_1 + \cdots + e_k} = b^e \bmod p$. Security follows from the observation that the exponent e is shared among the k peers using a (k, k)-threshold secret sharing scheme. Thus, the only way to reconstruct e is to collect all the k shares.

Computation with secure base and exponent. Using a similar idea, it is possible to compute b^e while keeping both the base b and the exponent e private. The main difference is that, in this case, we need to properly share both the base and the exponent and recombine the partial results.

Procedure SECURE_BASE_EXP(b, e, p, k^2)

- randomly select k^2 peers out of the t and let $\{ID_1, \ldots, ID_{k^2}\}$ be their identifiers
- pick $e_1, \ldots, e_{k-1} \in_R Z_{p-1}$
- pick $b_1, \ldots, b_{k-1} \in_R Z_p^\star$
- $e_k = e - (e_1 + \ldots + e_{k-1}) \bmod (p-1)$.
- $b_k = b / \prod_{i=1}^{k} b_i \bmod p$.
- $r_{i,j} = \text{MOD\_EXP}(b_i, e_j, p, ID_{i(j-1)+j-1})$, for $i = 1, \ldots, k$ and $j = 1, \ldots, k$
- $r_i = \prod_{j=1}^{k} r_{i,j} \bmod p$
- $r = \prod_{i=1}^{k} r_i \bmod p$
- output r

The correctness of the above procedure can be derived by observing that $r_i = b_i^e \bmod p$ since SECURE_BASE_EXP implicitly contains an invocation of the procedure SECURE_EXP with parameters b_i and e. Furthermore, $r = b_1^e \cdot \ldots \cdot b_k^e = (\prod_{i=1}^{k} b_i)^e = b^e \bmod p$. The security of the procedure derives from the fact that we use two independent (k, k)-threshold secret sharing schemes for sharing b and e. Since each pair (b_i, e_j), for $i, j \in \{1, \ldots, k\}$, is assigned to a different peer, the only way to reconstruct the value of b (resp., the value of e) is to collect all the values b_i (resp., e_i).

Distributing the Commitment Operations. Given the above procedures, we can distribute the computation of commitments as follows:

Recall that, given the public parameters (p, g, h), a soft commitment consists of a pair `Scom` $= (g^{r_0}, g^{r_1})$ and `Sdec` $= (r_0, r_1)$, where r_0 and r_1 are randomly chosen.

In this case, the values r_0 and r_1 need to be kept private since they are used for the teasing of `Scom`. Thus, the SoftCommit procedure can be distributed as follows:

Procedure SoftCommit(p, g, h, k)

```
- $r_0, r_1 \in_R Z_p$
- $y_0 = \text{Secure\_Exp}(g, r_0, p, k)$
- $y_1 = \text{Secure\_Exp}(g, r_1, p, k)$
- output $\texttt{Scom} = (y_0, y_1)$, and $\texttt{Sdec} = (r_0, r_1)$.
```

The correctness of the above procedure follows immediately by inspection, while its security follows from the fact that y_0 and y_1 are computed by independent executions of the Secure_Exp algorithm.
Let us now consider hard commitments. A hard commitment, given the public parameters (p, g, h) and a message m, consists of a pair $\texttt{com} = (g^m \cdot (h^{r_1})^{r_0}, h^{r_1})$ and $\texttt{dec} = (r_0, r_1)$ where r_0 and r_1 are randomly chosen. In this case, clearly the value m need to be private for guaranteeing the hiding property of the commitment. The exponents r_0 and r_1 need to be private since they constitute the decommitment key `dec`. Finally the value h^{r_1} needs to be kept private since it constitutes the second component of `com`. Indeed, if such value becomes public, an attacker that eavesdrops a pair (x, h^{r_1}) knows, w.h.p. that such pair defines a hard commitment, contradicting the indistinguishably of hard and soft commitments. Thus, the Commit procedure can be distributed as follows:

Procedure Commit(p, g, h, m, k^2)

```
- $r_0, r_1 \in_R Z_p$.
- $w = \text{Secure\_Exp}(g, m, p, k)$
- $y_1 = \text{Secure\_Exp}(h, r_1, p, k)$
- $y_0 = \text{Secure\_Base\_Exp}(y_1, r_0, p, k^2)$
- set $\texttt{com} = (wy_0, y_1)$ and $\texttt{dec} = (r_0, r_1)$
```

The correctness of the above algorithm can be verified by inspection while, as before, its security follows from the independence of the computations for w, y_1 and y_0.

Distributing the verification. We can now show the distribution of load on the User side. Notice that, the same observations above also apply to the verification procedures described below.

Procedure VerifyOpen$(p, g, h, (C_0, C_1), (r_0, r_1), m, k^2)$

```
- $\overline{c_1} = \text{Secure\_Exp}(h, r_1, p, k)$
- $\overline{g} = \text{Secure\_Exp}(g, m, p, k)$
- $\overline{c_2} = \text{Secure\_Base\_Exp}(c_1, r_0, p, k^2)$
- if $C_1 = \overline{c_1}$ and $C_0 = \overline{g} \cdot \overline{c_2}$ output "Verified" else output "Failure".
```

Procedure $\textsc{VerifyTease}(p, g, h, (C_0, C_1), \tau, m, k^2)$

```
- $\overline{c_1} = \textsc{Secure\_Base\_Exp}(C_1, \tau, p, k^2)$
- $\overline{g} = \textsc{Secure\_Exp}(g, m, p, k)$
- if $C_0 = \overline{g} \cdot \overline{c_1}$ output "Verified" else output "Failure".
```

4 Certified Information Access Via M1-CIA

In this section we describe how to implement the CIA functionality based on M1-CIA. This description resembles the one in [3]. We consider a simple database D associating to a key x a value $D(x) = v$. Let us assume that all keys have the same length ℓ. A reasonable choice is $\ell = 128$ since on one hand it allows to have a large key space and, at the same time, it is possible to use hash functions for reducing the key size to this small value while preserving collision freeness. The database D can thus be represented by a height-ℓ binary tree where leaf numbered x contains the value $v = D(x)$. If no value is associated by the database to key x, the leaf numbered x contains the special value $\perp$.

The CertifiedDBOwner constructs a binary tree that can be described as follows: Leaves of the tree contain the commitment of elements of the database. Each internal node of the tree contains the commitment to the contents of its two children. The commitment to the root of such a tree constitutes the public information that is sent to the PubInfoStorage.

To respond to a query about x, the CertifiedDBOwner simply decommits the corresponding leaf and provides the authenticating path (along with all the decommitments) to the root. The problem with this approach is that it requires time exponential in the height of the tree: if we choose $\ell = 128$, then 2^{128} commitments need to be computed.

This is where M1-CIA helps. Observe that the exponential-size tree has large empty subtrees (that is, subtrees where each leaf is a commitment to $\perp$). Instead of actually computing such a subtree ahead of time, the CertifiedDBOwner forms the root of this subtree as a soft commitment and does not do anything for the rest of the tree. Thus the size of the tree is reduced to at most $2\ell|D|$, where $|D|$ represents the number of element in the database. Responding to a query about x such that $D(x) \neq \perp$ is still done in the same way. If instead $D(x) = \perp$, the CertifiedDBOwner teases the path from the root to x. More precisely, the path from the root to x will consists of hard commitments until the root R of the empty subtree containing x is encountered. All hard commitments from the root of the tree to R are teased to their real values (recall that hard commitments can be teased only to their real value). Then, the CertifiedDBOwner generates a path of soft commitments from R to (the leaf with number) x ending with the commitment of $\perp$. Each soft commitment corresponding to a node along the path is teased to the soft commitments corresponding to its two children. The User simply needs to verify that each teasing has been correctly computed. We stress that for positive queries (that is, queries for x such that $D(x) \neq \perp$) the User expects to see *opening* of hard commitments whereas for negative queries (that is, queries for x such that $D(x) = \perp$) the User expects to see *teasing* of commitments; some of them will be hard commitments and some will be soft commitments but the user cannot say which ones are which. Due to space limitations, we describe the above procedures in details in Appendix A.

5 The Architectural Design Principles

In this section we briefly describe the architectural design for a system implementing the primitives described above. We identify four different entities, CertifiedDBOwner, User and PubInfoStorage and Peer.

The applications cooperate as follow. At startup, the PubInfoStorage generates the public parameters that will be used for the M1-CIA implementations. Since public parameters are used both for generating the public information and verifying all the answers to queries, such generation is carried out once, and the parameters are stored in a file.

At this point the CertifiedDBOwner generates the public information with the help of the Peer applications that is run on a set of peers. The public information is sent and stored by the PubInfoStorage. When the User queries the CertifiedDBOwner, he obtains a reply that is verified against the public information held by the PubInfoStorage. We assume point-to-point secure communication among CertifiedDBOwner, User and PubInfoStorage. Furthermore, the communication between any peer and an entity invoking its service is also secured.

In our experience, the most time-consuming operation is the creation of the public information executed by the CertifiedDBOwner. Notice that the distribution of load clearly helps in reducing the time required for such operation if the number t of available peers is bigger than k^2, where k is the security threshold.

A possible way of further reducing the computation time by using pre-computation. We observe that a soft commitment is composed by a pair (g^{r_0}, g^{r_1}) where both r_0 and r_1 are random values. Furthermore, a hard commitment is a pair whose second component is h^r where, again, r is a random value. Clearly it is immaterial whether or not the value r is chosen by the peer. What it does matter is that the CertifiedDBOwner, given the information obtained by the peers, is able to compute some pair $(r', g^{r'})$ (resp., $(r', h^{r'})$) where both components are private.

We can thus introduce a new service, the Batch_Mod_Exp that, given the public parameter held by the PubInfoStorage, computes and stores two lists of pairs, (r, g^r) and $(r', h^{r'})$. Whenever the CertifiedDBOwner needs to compute a soft (resp., hard) commitment, it may simply invoke the Batch_Mod_Exp service that will return the first pair (r, g^r) (resp., (r, h^r)) and remove it from the list.

The pre-computation technique just described can be extremely useful in the case in which the public information associated to the database need to be recomputed frequently. As stated in the introduction, there are no currently available *efficient* solution for implementing a CIA service in case the database is dynamic. Consider, for example, the case in which the database needs to be modified, say, once per day. The only way of guaranteeing the security of the service is to recompute the public information each time. In this case the company where the CertifiedDBOwner is running, may set up peers on its computers so that they pre-compute the information when they are not used, e.g., at night.

6 Conclusion

In this paper we have presented a distributed architecture for a Certified Information Access system. We have shown that it is possible to securely distribute the load of the most time-consuming operations among a set of untrusted peers. Furthermore, we have presented a solution that allows the usage of pre-computation in order to reduce the time needed by the CertifiedDBOwner for generating the public information.

References

1. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004)*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.
2. Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In Shai Halevi and Tal Rabin, editors, *Third Theory of Cryptography Conference (TCC 2006)*, volume 3876 of *Lecture Notes in Computer Science*, pages 120–144. Springer, 2006.
3. Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to Zero-Knowledge sets. In Ronald Cramer, editor, *24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2005)*, volume 3494 of *Lecture Notes in Computer Science*, pages 422–439. Springer, 2005.
4. Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. In *TR-CS0917, Department of Computer Science, Technion,*, 1997.
5. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.
6. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *38th Symposium on Foundations of Computer Science (FOCS 1997)*, pages 364–373. IEEE Computer Society, 1997.
7. Moses Liskov. Updatable zero-knowledge databases. In Bimal K. Roy, editor, *11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2005)*, volume 3788 of *Lecture Notes in Computer Science*, pages 174–198. Springer, 2005.
8. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 80–91. IEEE Computer Society, 2003.

9. Sanjeev Kumar Mishra and Palash Sarkar. Symmetrically private information retrieval. In Bimal K. Roy and Eiji Okamoto, editors, *First International Conference in Cryptology in India (Indocrypt 2000)*, volume 1977 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 2000.
10. Wakaha Ogata and Kaoru Kurosawa. Oblivious keyword search. *Journal of Complexity*, 20(2-3):356–371, 2004.
11. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

A Certified Information Access Via M1-CIA

In this appendix we describe in more details the generation of the tree of commitments, the construction of a answer by the CertifiedDBOwner and the verification procedure executed by the User.

Generation of Public Information. To generate the public information representing the database D, the CertifiedDBOwner proceeds as follows. We stress, that even though not explicitly specified, all calls to Commit and SoftCommit take as input also the public parameter generated by the PubInfoStorage during the SETUP.

The construction of the tree of commitments starts from its leaves. More precisely, for each x such that $D(x) \neq \perp$, the CertifiedDBOwner produces $(C_x, D_x) = \text{Commit}(D(x))$. Then for each x such that $D(x) = \perp$ but $D(x') \neq \perp$ (where x' is x with the last bit flipped), the CertifiedDBOwner produces $(C_x, D_x) = \text{SoftCommit}$. For all the others x, $C_x = \emptyset$. Now the tree is constructed in a bottom-up fashion as follows: for each level $i = \ell - 1, \cdots, 0$ and for each string s of length i, define C_s as follows:

1. If $C_{s0} \neq \emptyset$ and $C_{s1} \neq \emptyset$, the let $(C_s, D_s) = \text{Commit}((C_{s0}, C_{s1}))$.
2. For all s such that $C_{s'}$ has been defined in the previous step (s' is s with the last bit flipped) but C_s has not, define $(C_s, D_s) = \text{SoftCommit}$.
3. For all other s, define $C_s = \emptyset$.

The value at the root C_ϵ is the public information. If we have $C_\epsilon = \emptyset$ we set $(C_\epsilon, D_\epsilon) = \text{SoftCommit}$.

Answer to a query. This method constructs an object that contains the answer to the query and a proof of validity composed by the decommitments of the corresponding leaf along with the authenticating path (together with all the decommitments) to the root.

More specifically, we distinguish between the case in which the query x is such that $D(x) \neq \perp$ and $D(x) = \perp$. For a string x we denote by $x|_i$ the first i bits of x and by $(x|_i)'$ the first $i-1$ bits of x followed by the i-th bit of x flipped.

If $D(x) \neq \perp$, the authenticating path is computed by sending $D(x)$, the corresponding decommitment key D_x and, for $0 \leq i \leq \ell - 1$, the values $(C_{x|_i 0}, C_{x|_i 1})$ along with the decommitment key $D_{x|_i}$.

Suppose instead that $D(x) = \perp$ and let h be largest value such that $C_{x|_h} \neq \emptyset$, set $(C_x, D_x) = \textsc{Commit}(\perp)$, and build a path from x to $C_{x|_h}$ as follows: set $(C_{x'}, D_{x'}) = \textsc{SoftCommit}$; for each level i from $\ell - 1$ to $h + 1$, define $(C_{x|_i}, D_{x|_i}) = \textsc{Commit}(C_{x|_i0}, C_{x|_i1})$, and $(C_{(x|i)'}, D_{(x|i)'}) = \textsc{SoftCommit}$. Note that the only values inside the tree redefined by the above procedure are those that were not defined before.

Let $\tau_x = \textsc{Tease}(D(x), C_x, D_x)$ and $\tau_{x|_i} = \textsc{Tease}((C_{x|_i0}, C_{x|_i1}), C_{x|_i}, D_{x|_i})$ for $0 \leq i < \ell$. The response to the query consists of $\perp$ along with its validation path: $(C_{x|_i}, C_{(x|_i)'})$ for $1 \leq i \leq \ell$ and $\tau_{x|_i}$ for $0 \leq i \leq \ell$.

Verification of an Answer. To verify the certified answer, for a query to a key x such that $D(x) \neq \perp$, User executes the VerifyOpen method on all the decommitments received, from the bottom up to the root. The last verification is made against the database commitment that he has previously retrieved from the PubInfoStorage. In case the key that has been queried is not in the database, then the User has to execute VerifyTease instead of VerifyOpen.

Hashing the values. In the discussion above, we have in several points constructed a hard commitment of two hard commitments. This will make the size of the commitment roughly double at each level. Instead we assume that instead of committing to a string (or to a pair of strings) we commit to its hash value computed using a collision-resistant hash function H which hashes down a string to a fixed length ℓ. Notice, again, that $\ell = 128$ is a good choice. Similarly, we can have a database with keys of different length if, instead of storing the pair (x, v) we store the pair $(H(x), v)$.

Linkability of Some Blind Signature Schemes[⋆]

Swee-Huay Heng[1], Wun-She Yap[2], and Khoongming Khoo[3]

[1] Centre for Cryptography and Information Security (CCIS)
Faculty of Information Science and Technology
Multimedia University, Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

[2] Centre for Cryptography and Information Security (CCIS)
Faculty of Engineering
Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
wsyap@mmu.edu.my

[3] DSO National Laboratories
20 Science Park Drive, Singapore 118230
kkhoongm@dso.org.sg

Abstract. Unforgeability and blindness are two important properties of blind signature. The latter means that after interacting with various users, the signer is unable to link a valid message-signature pair. In ICCSA 2006, Zhang *et al.* showed that a signer in an identity-based blind signature scheme proposed by Huang *et al.* is able to link a valid message-signature pair obtained by some user. They also presented an improved scheme to overcome this flaw. In ICICIC 2006, Zhang and Zou showed that the identity-based blind signature scheme proposed by Zhang and Kim also suffered from the similar linkability attack. In this paper, we first show that the so-called linkability can be shown for Zhang *et al.* scheme as well. We then point out that the linkability attack against the Huang *et al.* scheme and the Zhang-Kim scheme is invalid.

Keywords: Blind signature, identity-based, linkability, blindness.

1 Introduction

The concept of blind signatures was first introduced by Chaum [3] in 1982. A blind signature scheme is an interactive two-party protocol between a user and a signer. Informally, a blind signature is a signature scheme that incorporates a signing protocol that allows the signer to sign a document submitted by a user blindly, without obtaining any information about the document itself. This cryptographic scheme provides anonymity of users and is especially suited for use in e-cash and e-voting systems.

On the other hand, identity (ID)-based public key cryptography is a concept formalized by Shamir in 1984 [6]. In ID-based schemes, users need exchange

[⋆] The first and second authors gratefully acknowledge the Malaysia IRPA grant (04-99-01-00003-EAR) and e-Science fund (01-02-01-SF0032).

neither private keys nor public keys. Generally, an ID-based scheme is an asymmetric system wherein the public key is effectively replaced by or constructed from a user's publicly available identity information (e.g., name, email address, IP address) which uniquely identifies the user and can be undeniably associated with the user. The services of a trusted third party called private key generator (PKG) are needed solely to generate private keys for users using the PKG's master-key and the user's public identity information. The main technical difference between ID-based cryptography and the traditional public key infrastructure (PKI) systems using certificates is in the binding between the public and private keys and the means of those keys. In a traditional PKI, this is achieved through the use of a certificate.

The first ID-based blind signature (IBBS) scheme was put forth by Zhang and Kim in 2002 [7]. Later, the same authors provided an improved IBBS scheme [8]. Unlike the first scheme, they claimed that the general parallel attack of this improved scheme does not depend on the difficulty of ROS-problem, this was then falsified by Huang *et al.* [4]. Huang *et al.* showed that the security against generic parallel attack of Zhang and Kim's improved scheme [8] still depends on the difficulty of ROS-problem. Huang *et al.* [4] further proposed another scheme which offers advantages in runtime, communication and memory requirements over the first two schemes.

In ICCSA 2006, Zhang *et al.* showed that a signer in an IBBS scheme proposed by Huang *et al.* is able to link a valid message-signature pair obtained by some user [9]. They also presented an improved scheme to overcome this flaw. Recently, in ICICIC 2006, Zhang and Zou also showed that the identity-based blind signature scheme proposed by Zhang and Kim [7] is vulnerable to the same linkability attack [10]. In this paper, we first show that the so-called *linkability* can be shown for Zhang *et al.* scheme as well. We then show that the *linkability* attack is invalid. We also compare the performance between the Zhang-Kim scheme, the Zhang *et al.* scheme and the Huang *et al.* scheme. From the analysis, we can see that the Huang *et al.* scheme is more efficient than the Zhang *et al.* scheme.

In Section 2, we review some preliminaries. In Section 3, we review the Huang *et al.* and the Zhang-Kim IBBS scheme. In Section 4, we review the Zhang *et al.* scheme and discuss the linkability issue on the Zhang *et al.* scheme before falsifying the soundness of the linkability attack claimed by Zhang *et al.* against the Huang *et al.* scheme. Finally, we conclude this paper in Section 5.

2 Preliminaries

2.1 Bilinear Pairings

Throughout this paper, $(G_1, +)$ and $(G_2, \cdot)$ denote two cyclic groups of prime order q. A *bilinear map*, $e : G_1 \times G_1 \rightarrow G_2$ satisfies the following properties:

1. Bilinearity: For all $P, Q, R \in G_1$, $e(P+Q, R) = e(P, R)e(Q, R)$ and $e(P, Q+R) = e(P, Q)e(P, R)$.

2. Non-degeneracy: $e(P, Q) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

2.2 Identity-Based Blind Signature

An identity-based blind signature (IBBS) scheme is considered as the combination of a general blind signature scheme and an ID-based one. In other words, it is a blind signature but the public key used in the verification is the signer identity such that no certificate is needed in authenticating the signer's public key. Now we review the framework and security model of an IBBS scheme [5,1,7,8,4].

An IBBS scheme is a digital signature scheme which involves three parties: a trusted third party called the PKG, a signer and a user. It consists of the following four algorithms:

1. `Setup` is a probabilistic polynomial-time (PPT) algorithm run by the PKG that takes a security parameter k and returns the system parameters *params* and *master-key*.
2. `Extract` is a deterministic algorithm run by the PKG that takes *params*, *master-key* and an entity identifier ID $\in \{0,1\}^*$ as input. It returns the signer private key S_{ID}.
3. `Issue` is an interactive PPT signature issuing protocol between a signer and a user. Suppose that the user is given its input tape (ID, m) where m is a message and the signer is given its input tape (ID, S_{ID}). The signer and the user then engage in the signature issuing protocol. At the end of this protocol, the signer outputs either "completed" or "non-completed" while the user outputs either $\perp$ or the signature σ of the message m.
4. `Verify` is a deterministic polynomial-time algorithm that accepts a signature σ, message m, *params* and ID and outputs *true* if the signature is correct, or $\perp$ otherwise.

These algorithms must satisfy the standard consistency constraint of an ID-based blind signature, i.e. if $\sigma =$ `Issue`(m, ID, S_{ID},*params*), `Verify`(σ, m, ID, *params*) $=$ *true* must hold.

A secure ID-based blind signature should have the property of blindness and the unforgeability against adaptive chosen message and ID attacks. We provide the definition for the former only since we are particularly dealing with this notion in this paper.

Definition 1 ***(Blindness).*** *Let $\mathcal{A}$ be the Signer or a PPT algorithm that controls the Signer. $\mathcal{A}$ is involved in the following game with two honest users, namely U_0 and U_1.*

1. $(ID, S_{ID}) \leftarrow$ `Extract`$(params, S_{ID})$.
2. $(m_0, m_1) \leftarrow \mathcal{A}(ID, S_{ID})$ *($\mathcal{A}$ produces two messages).*
3. *Select $b \in \{0,1\}$. Put m_b and m_{1-b} to the read-only input tape of U_0 and U_1 respectively.*

4. $\mathcal{A}$ engages in the signature issuing protocol with U_0 and U_1 in an arbitrary order.
5. If U_0 and U_1 output $\sigma(m_b)$ and $\sigma(m_{1-b})$ respectively using their private tapes, then give those outputs to $\mathcal{A}$. Otherwise, give $\perp$ to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

We say that $\mathcal{A}$ wins the game if $b' = b$. An IBBS is blind if there is no PPT algorithm $\mathcal{A}$ that wins the game with probability at least $1/2 + 1/k^c$ for any constant $c > 0$. The probability is taken over the coin flips of `Extract`, *U_0, U_1 and $\mathcal{A}$.*

3 The Huang *et al.* and the Zhang-Kim IBBS Schemes

3.1 The Huang *et al.* IBBS Scheme

1. `Setup`: Choose a group G_1 which is a cyclic additive group generated by P with prime order q. Choose a cyclic multiplicative group G_2 with the same order q and a bilinear pairing $e : G_1 \times G_1 \to G_2$. Pick a random $s \in Z_q^*$ and set $P_{pub} = sP$. Choose two cryptographic hash functions $H_1 : \{0,1\}^* \times G_2 \to Z_q^*$ and $H_2 : \{0,1\}^* \to G_1$. Publicize the system parameters $params = (G_1, G_2, e, q, P, P_{pub}, H_1, H_2)$ and keep the master key s secret.
2. `Extract`: Given an identity ID, compute $P_{ID} = H_2(ID)$ and return the corresponding private key $S_{ID} = sP_{ID}$.
3. `Issue`: The user first chooses $P_1 \in G_1$ and computes $e(P_1, P)$ beforehand. In order to get a signature on a message m, the interaction between the user and the signer is as follows:
   - `Sign (Part 1)`: The signer randomly chooses $r \in Z_q^*$ and computes $R' = e(P_{ID}, P_{pub})^r$ before sending R' to the user as the commitment.
   - `Blinding`: The user randomly chooses $t_1, t_2 \in Z_q^*$ as blinding factors and computes $R = R'^{t_1} e(P_1, P)^{t_2}$, $h = H_1(m, R)$ and $h' = ht_1$ before sending h' to the signer as the challenge.
   - `Sign (Part 2)`: The signer sends back V' to the user as the response where $V' = (rh' + 1)S_{ID}$.
   - `Unblinding`: The user checks whether $e(V', P) = R'^{h'} e(P_{ID}, P_{pub})$. If yes, then the user computes $V = V' + ht_2 P_1$ and outputs the signature $\sigma = (R, V)$.
4. `Verify`: To verify a signature $\sigma = (R, V)$ on a message m for ID, the verifier checks whether $e(V, P) = R^{H_1(m,R)} e(P_{ID}, P_{pub})$.

3.2 The Zhang-Kim IBBS Scheme

1. `Setup`: The same as Section 3.1.
2. `Extract`: The same as Section 3.1.
3. `Issue`:
   - `Sign (Part 1)`: The signer randomly chooses $r \in Z_q^*$ and computes $R = rP$ before sending R to the user as the commitment.

- `Blinding`: The user randomly chooses $a, b \in Z_q^*$ as blinding factors and computes $t = e(bQ_{ID} + R + aP, P_{pub})$ and $c' = H_1(m, t) + b$ before sending c' to the signer as the challenge.
- `Sign (Part 2)`: The signer sends back V' to the user as the response where $V' = c'S_{ID} + rP_{pub}$.
- `Unblinding`: The user computes $V = V' + aP_{pub}$ and $c = c' - b$ and outputs the signature $\sigma = (V, c)$.

4. `Verify`: To verify a signature $\sigma = (V, c)$ on a message m for ID, the verifier checks whether $c = H_1(m, e(V, P)e(Q_{ID}, P_{pub})^{-c})$.

4 Soundness of the Linkability Attack

Recently, in ICCSA 2006, Zhang *et al.* claimed that the Huang *et al.* blind signature [4] did not satisfy the blindness by analyzing the security of the scheme where the signer is able to link a valid message-signature pair obtained by some user after interacting with various users [9]. In ICICIC 2006, Zhang and Zou also showed that the Zhang-Kim IBBS scheme [7] is vulnerable to the same linkability attack [10]. In this section, we first review the Zhang *et al.* scheme. We then review the Zhang *et al.* attack on the Huang *et al.* IBBS scheme and the Zhang-Zou attack on the Zhang-Kim IBBS scheme. Subsequently, we show that this so-called *linkability* attack can also be applied to the Zhang *et al.* scheme [9]. Finally, we prove that the so-called *linkability* attack is in fact invalid.

4.1 The Zhang *et al.* IBBS Scheme

The Zhang *et al.* scheme [9] is considered as an allegedly improved scheme over the Huang *et al.* which served as the countermeasure against the linkability attack mounted by Zhang *et al.* against the latter.

1. `Setup`: The same as Section 3.1.
2. `Extract`: The same as Section 3.1.
3. `Issue`: The user first chooses $P_1 \in G_1$ and computes $e(P_1, P)$ beforehand. In order to get a signature on a message m, the interaction between the user and the signer is as follows:
   - `Sign (Part 1)`: The signer randomly chooses $r \in Z_q^*$ and computes $R' = e(P_{ID}, P_{pub})^r$ before sending R' to the user as the commitment.
   - `Blinding`: The user randomly chooses $t_1, t_2, t_3 \in Z_q^*$ as the blinding factors and computes $R = R'^{t_1} e(P_{ID}, P_{pub})^{t_1 t_2} e(P_1, P)^{t_3}$, $h = H_1(m, R)$ and $h' = ht_1^{-1} + t_2$ before sending h' to the signer as the challenge.
   - `Sign (Part 2)`: The signer sends back V' to the user as the response where $V' = (r + h')S_{ID}$.
   - `Unblinding`: The user computes $V = t_1 V' + t_3 P_1$ and outputs the signature $\sigma = (R, V)$.
4. `Verify`: To verify a signature $\sigma = (R, V)$ on a message m for ID, the verifier checks whether $e(V, P) = R \cdot e(P_{ID}, P_{pub})^{H_1(m,R)}$.

4.2 Linkability of the Huang *et al.* IBBS Scheme

We briefly review the Zhang *et al.* attack below. During the interactive protocol execution between the signer and the user, the transcript (R', h', V') is generated. Given a blind signature $\sigma = (R, V)$ on a message m, the signer executes the following steps:

1. Compute $\alpha = e(V - V', P)$.
2. Compute $\beta = R'^{h'}$.
3. Compute $h = H_1(m, R)$ and check whether $\alpha \cdot \beta = R^h$. If equal, then it indicates that the signer is managed to link the message-signature pair.

Since $V = V' + ht_2P_1$, thus the signer computes α as follows:

$$\begin{aligned}\alpha &= e(V - V', P)\\ &= e(ht_2P_1, P)\\ &= e(P_1, P)^{ht_2}\end{aligned}$$

The signer manages to compute $\beta = R'^{h'}$ since $h' = ht_1$ is known. Finally, $\alpha \cdot \beta$ is computed as follows:

$$\begin{aligned}\alpha \cdot \beta &= e(P_1, P)^{ht_2} \cdot R'^{ht_1}\\ &= \{e(P_1, P)^{t_2} \cdot R'^{t_1}\}^h\\ &= R^h \text{ where } R = R'^{t_1} e(P_1, P)^{t_2})\end{aligned}$$

Thus, Zhang *et al.* claimed that the Huang *et al.* IBBS scheme [4] has no blindness.

4.3 Linkability of the Zhang-Kim IBBS Scheme

Zhang and Zou showed an attack on the Zhang-Kim IBBS scheme [7]. We briefly review the Zhang and Zou attack now. During the interactive protocol execution between the signer and the user, the transcript (R, c', V') is generated. Given a blind signature $\sigma = (c, V)$ on a message m, the signer executes the following steps:

1. Compute $\alpha = e(V - V', P)$.
2. Compute $\beta = c' - c$.
3. Compute $\delta = e(R, P_{pub})$.
4. Compute $t' = \alpha \cdot \delta \cdot e(Q_{ID}, P_{pub})^{\beta}$.
5. Check whether $c = H_1(m, t')$.

Notice that

$$\begin{aligned}t' &= \alpha \cdot \delta \cdot e(Q_{ID}, P_{pub})^{\beta}\\ &= e(V - V', P) \cdot e(R, P_{pub}) \cdot e(Q_{ID}, P_{pub})^{(c'-c)}\\ &= e(aP_{pub}, P) \cdot e(R, P_{pub}) \cdot e(Q_{ID}, P_{pub})^{b}\\ &= e(aP + R + bQ_{ID}, P_{pub})\\ &= t\end{aligned}$$

Thus, we have that the relation $H_1(m, t') = H_1(m, t) = c$ holds and it means that the signer is able to link a message-signature pair. Zhang and Zou then claimed that the Zhang-Kim IBBS scheme has no blindness as well.

4.4 Linkability of the Zhang *et al.* Scheme

Now, we show that the similar so-called *linkability* can be shown for the Zhang *et al.* scheme [9] as well.

During the interactive protocol execution between the signer and the user, the transcript (R', h', V') is generated. Given a blind signature $\sigma = (R, V)$ on a message m, the signer executes the following steps:

1. Compute $\alpha = e(V - V', P)$.
2. compute $h = H_1(m, R)$ and set $\beta = h - h'$.
3. Compute $t' = \alpha \cdot e(P_{ID}, P_{pub})^{\beta} \cdot R'$
4. Check whether $h = H(t', m)$. If equal, then it indicates that the signer is managed to link the message-signature pair.

Since $V = t_1V' + t_3P_1$, the signer can compute α as follows:

$$\begin{aligned}
\alpha &= e(V - V', P) \\
&= e(t_1V' + t_3P_1 - V', P) \\
&= e((t_1 - 1)V' + t_3P_1, P) \\
&= e((t_1 - 1)V', P)e(t_3P_1, P) \\
&= e((t_1 - 1)(r + h')S_{ID}, P)e(t_3P_1, P) \\
&= e(P_{ID}, P_{pub})^{(t_1-1)(r+h')}e(P_1, P)^{t_3} \\
&= e(P_{ID}, P_{pub})^{(rt_1-r+t_1h'-h')}e(P_1, P)^{t_3}
\end{aligned}$$

β can be computed as $h' - h$ since $h' = ht_1^{-1} + t_2$ is known. Finally, t' is computed as follows:

$$\begin{aligned}
t' &= \alpha \cdot e(P_{ID}, P_{pub})^{\beta} \cdot R' \\
&= e(P_{ID}, P_{pub})^{(rt_1-r+t_1h'-h')}e(P_1, P)^{t_3}e(P_{ID}, P_{pub})^{\beta}R' \\
&= e(P_{ID}, P_{pub})^{(rt_1-r+t_1(ht_1^{-1}+t_2)-h')}e(P_1, P)^{t_3}e(P_{ID}, P_{pub})^{h'-h}e(P_{ID}, P_{pub})^{r} \\
&= e(P_{ID}, P_{pub})^{(rt_1-r+h+t_1t_2-h')}e(P_1, P)^{t_3}e(P_{ID}, P_{pub})^{h'-h}e(P_{ID}, P_{pub})^{r} \\
&= e(P_{ID}, P_{pub})^{(rt_1+t_1t_2)}e(P_1, P)^{t_3} \\
&= e(P_{ID}, P_{pub})^{rt_1}e(P_{ID}, P_{pub})^{t_1t_2}e(P_1, P)^{t_3} \\
&= R_1'^{t}e(P_{ID}, P_{pub})^{t_1t_2}e(P_1, P)^{t_3} \\
&= R
\end{aligned}$$

Thus, we have $h = H_1(m, t')$. Assuming that the linkability attack shown by Zhang *et al.* and Zhang-Zhou is sound, then the Zhang *et al.* scheme which is an improvement over the Huang *et al.* scheme has no blindness too.

4.5 Soundness of the Linkability Attack

At first glance, it seems that Zhang *et al.*'s claim is true. Nevertheless, we are going to show that their claim is wrong. The main reason is that this proposed attack works even if the blind signature is not generated from the protocol, meaning that even if there is totally no connection between the signature and the protocol transcript.

Let $\mathcal{A}$ be the signer or a PPT algorithm that controls the signer. $\mathcal{A}$ is involved in the *blindness* game with two honest users, namely U_0 and U_1. First, $b \in \{0, 1\}$ is selected randomly. $\mathcal{A}$ engages in the `Issue` protocol with U_0 and U_1 in an arbitrary order. Assume that U_0 and U_1 output $\sigma(m_b)$ and $\sigma(m_{1-b})$ respectively using their private tape, and give those outputs to $\mathcal{A}$. The output of the `Issue` protocol can be seen as in Table 1.

Table 1. Output of the Issue Protocol

	U_0	U_1
Transcript	(R'_0, h'_0, V'_0)	(R'_1, h'_1, V'_1)
Resulting message-signature pair	(m_0, R_0, V_0)	(m_1, R_1, V_1)

Now, assume that $\mathcal{A}$ has the knowledge of (R'_0, h'_0, V'_0) and it wants to link the transcript with the output of U_1: $\sigma(m_1) = (R_1, V_1)$ in order to ensure the so-called *linkability*. We apply the Zhang *et al.* attack to show that the linkability algorithm always returns true even if the blind signature has totally no connection with the protocol transcript, thus we prove that $\mathcal{A}$ is unable to derive a link between a protocol view and a blind signature that has no relationship with the protocol view. This can be exhibited as follows:

1. Let $V'_0 = (r_0 h'_0 + 1)S_{ID}$ and $V_1 = V'_1 + h_1 t_2 P_1 = (r_1 h'_1 + 1)S_{ID} + h_1 t_2 P_1$.
2. Compute α as follows:

$$\begin{aligned}
\alpha &= e(V_1 - V'_0, P)\\
&= e(\{(r_1 h'_1 + 1)S_{ID} + h_1 t_2 P_1\} - (r_0 h'_0 + 1)S_{ID}, P)\\
&= e(r_1 h'_1 S_{ID} + h_1 t_2 P_1 - r_0 h'_0 S_{ID}, P)\\
&= e((r_1 h'_1 - r_0 h'_0)S_{ID} + h_1 t_2 P_1, P)\\
&= e((r_1 h'_1 - r_0 h'_0)S_{ID}, P)e(h_1 t_2 P_1, P)\\
&= e(P_{ID}, P_{pub})^{(r_1 h'_1 - r_0 h'_0)} e(P_1, P)^{h_1 t_2}
\end{aligned}$$

3. Compute β as follows:

$$\beta = R_0'^{h'_0} = e(P_{ID}, P_{pub})^{r_0 h'_0}$$

4. Compute $\alpha \cdot \beta$ as follows:

$$\begin{aligned}\alpha \cdot \beta &= e(P_{ID}, P_{pub})^{(r_1 h'_1 - r_0 h'_0)} e(P_1, P)^{(h_1 t_2)} \cdot e(P_{ID}, P_{pub})^{r_0 h'_0} \\ &= e(P_{ID}, P_{pub})^{(r_1 h'_1 - r_0 h'_0 + r_0 h'_0)} e(P_1, P)^{h_1 t_2} \\ &= e(P_{ID}, P_{pub})^{r_1 h_1 t_1} e(P_1, P)^{h_1 t_2} \\ &= \{e(P_{ID}, P_{pub})^{r_1 t_1} e(P_1, P)^{t_2}\}^{h_1} \\ &= R_1^{h_1}\end{aligned}$$

where $h'_1 = h_1 t_1$ and $R_1 = R_1'^{t_1} e(P_1, P)^{t_2} = e(P_{ID}, P_{pub})^{(r_1 t_1)} e(P_1, P)^{t_2}$

Based on the above computation, the linking algorithm always returns true and thus this shows that (R'_0, h'_0, V'_0) can be linked with (m_1, R_1, V_1). Hence, the proposed attack of Zhang et al. [9] is invalid. The similar analysis applies to the linkability attack on the Zhang-Kim scheme [7] and the Zhang *et al.* scheme [9].

4.6 A Comparison

We give a comparison between the Zhang-Kim scheme [7], the Huang *et al.* [4] and the Zhang *et al.* [9] IBBS schemes in terms of their computational complexity. We denote BP as the bilinear pairing operation, PM as the point multiplication on G_1, PA as the point addition on G_1 and E as the exponentiation on G_2. The result is summarized in Table 2.

Table 2. A Comparison

Scheme	`Issue`	`Verify`
Zhang-Kim [7]	$1BP + 6PM + 4PA$	$2BP + 1E$
Huang *et al.* [4]	$2PM + 1PA + 4E$	$1BP + 1E$
Zhang *et al.* [9]	$3PM + 1PA + 4E$	$1BP + 1E$

It can be easily seen that the original Huang *et al.* scheme [4] is more efficient than the Zhang *et al.* scheme [9].

5 Conclusion

We falsified the *linkability* attack shown on the Huang *et al.* and the Zhang-Kim IBBS schemes by Zhang *et al.*, and Zhang and Zou respectively. Thus, the claim that the Huang *et al.* and the Zhang-Kim schemes have no blindness is wrong. Besides, we also compared the efficiency of the Zhang-Kim scheme, Huang *et al.* scheme and the Zhang *et al.* scheme. Based on our analysis, the Huang *et al.* scheme is the most efficient scheme.

References

1. M. Abe and T. Okamoto. Provably Secure Partially Blind Signature. *In Proceedings of CRYPTO 2000*, LNCS 1880, pp. 271-286, Springer-Verlag, 2000.
2. J. Cha and J. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. *In Proceedings of PKC 2003*, LNCS 2567, pp. 18-30, Springer-Verlag, 2003.
3. D. Chaum. Blind signatures for untraceable payments. *In Proceedings of CRYPTO 1982*, pp. 199-203, Springer-Verlag, 1983.
4. Z. Huang, K. Chen, Y. Wang. Efficient Identity-Based Signatures and Blind Signatures. *In Proceedings of CANS 2005*, LNCS 3574, pp. 120-133, Springer-Verlag, 2005.
5. D. Pointcheval and J. Stern. Provable Secure Blind Signature Schemes. *In Proceedings of ASIACRYPT 1996*, LNCS 1163, pp. 252-263, Springer-Verlag, 1996.
6. A. Shamir. Identity Based Cryptosystems and Signature Scheme. *In Proceedings of CRYPTO 1984*, LNCS 196, pp. 47-53, Springer-Verlag, 1984.
7. F. Zhang and K. Kim. ID-Based Blind Signature and Ring Signature from Pairings. *In Proceedings of ASIACRYPT 2002*, LNCS 2501, pp. 533-547, Springer-Verlag, 2002.
8. F. Zhang and K. Kim. Efficient ID-Based Blind Signature and Proxy Signature. *In Proceedings of ACISP 2003*, LNCS 2727, pp. 312-323, Springer-Verlag, 2003.
9. J. Zhang, T.Wei, J. Zhang and W. Zou. Linkability of a Blind Signature Scheme and Its Improved Scheme. *In Proceedings of ICCSA 2006*, LNCS 3983, pp. 262-270, Springer-Verlag, 2006.
10. J. Zhang and W. Zou. Linkability of a Blind Signature Scheme. *In Proceedings of ICICIC 2006*, Vol. 1, pp. 468-471, IEEE, 2006.

Optimistic Non-repudiation Protocol Analysis

Judson Santiago and Laurent Vigneron*

LORIA – Nancy Université
laurent.vigneron@loria.fr

Abstract. Non-repudiation protocols with session labels have a number of vulnerabilities. Recently Cederquist, Corin and Dashti have proposed an optimistic non-repudiation protocol that avoids altogether the use of session labels. We have specified and analysed this protocol using an extended version of the AVISPA Tool and one important fault has been discovered. We describe the protocol, the analysis method, show two attack traces that exploit the fault and propose a correction to the protocol.

1 Introduction

While security issues such as secrecy and authentication have been studied intensively [11], most interest in non-repudiation protocols has only come in recent years, notably in the yearly 1990s with the explosion of Internet services and electronic transactions.[1]

Non-repudiation protocols must ensure that when two parties exchange information over a network, neither one nor the other can deny having participated to this communication. Consequently a non-repudiation protocol must generate evidences of participation to be used in case of a dispute. With the advent of digital signatures and public key cryptography, the base for non-repudiation services was created. Given an adequate public key infrastructure, one having a signed message has an evidence of the participation and the identity of his party [7].

While non-repudiation can be provided by standard cryptographic mechanisms like digital signatures, fairness is more difficult to achieve: no party should be able to reach a point where they have the evidence or the message they require without the other party also having their required evidence. Fairness is not always required for non-repudiation protocols, but it is usually desirable.

A variety of protocols has been proposed in the literature to solve the problem of fair message exchange with non-repudiation. The first solutions were based on a gradual exchange of the expected information [7]. However this simultaneous secret exchange is troublesome for actual implementations because fairness is

* This work is supported by the ACI Sécurité SATIN and the RNTL project 03V360 Prouvé.

[1] See http://www.lsv.ens-cachan.fr/~kremer/FXbib/references.php for a detailed list of publications.

based on the assumption of equal computational power on both parties, which is very unlikely in a real world scenario. A possible solution to this problem is the use of a trusted third party (TTP), and in fact it has been shown that it is impossible to achieve fair exchange without a TTP [10,9]. The TTP can be used as a delivery agent to provide simultaneous share of evidences. The Fair Zhou-Gollmann protocol [16] is the most known example of non-repudiation protocol, using a TTP as a delivery agent of a key for decrypting the message sent by one agent to another agent; a significant amount of work has been done over this protocol and its derivations [2,6,13,17]. However, instead of passing the complete message through the TTP and thus creating a possible bottleneck, recent evolution of these protocols resulted in efficient, *optimistic* versions, in which the TTP is only involved in case anything goes wrong. Resolve and abort sub-protocols must guarantee that every party can complete the protocol in a fair manner and without waiting for actions of the other party (timeliness).

One of these recent protocols, which we describe in the following section, is the optimistic Cederquist-Corin-Dashti (CCD) non-repudiation protocol [3]. The CCD protocol has the advantage of not using session labels, contrariwise to many others in the literature [7,8,16,13]. A session label typically consists of a hash of all message components. Gürgens et al. [6] have shown a number of vulnerabilities associated to the use of session labels and, to our knowledge, the CCD protocol is the only optimistic non-repudiation protocol that avoids altogether the use of session labels.

In this paper we describe the CCD non-repudiation protocol, present the analysis method and explain two attack traces of an important flaw discovered in this protocol. The attack has been found after the specification and analysis of the protocol in the AVISPA Tool [1][2], using an extended version of the AtSe engine [15] that supports non-repudiation analysis. We propose a correction for the CCD protocol that have been successfully analysed for many scenarios.

2 The CCD Protocol

The CCD non-repudiation protocol has been created for permitting an agent A to send a message M to agent B in a fair manner. This means that agent A should get an evidence of receipt of M by B (EOR) if and only if B has really received M and the evidence of origin from A (EOO). EOR permits A to prove that B has received M, while EOO permits B to prove that M has been sent by A. The protocol is divided into three sub-protocols: the main protocol, an *abort* sub-protocol and a *resolve* sub-protocol.

2.1 Definition of the Main Protocol

This main protocol describes the sending of M by A to B and the exchange of evidences in the case where both agents can complete the entire protocol. If a

[2] `http://www.avispa-project.org`

problem happens to one of the agents, in order to finish properly the protocol, the agents can exchange messages with a trusted third party (TTP) by executing the *abort* or the *resolve* sub-protocol.

The main protocol is therefore composed of the following messages exchanges, described in the Alice&Bob notation:

1. $A \rightarrow B:\ \{M\}_K.EOO_M$ where $EOO_M = \{B.TTP.H(\{M\}_K).\{K.A\}_{Kttp}\}_{inv(Ka)}$
2. $B \rightarrow A:\ EOR_M$ where $EOR_M = \{EOO_M\}_{inv(Kb)}$
3. $A \rightarrow B:\ K$
4. $B \rightarrow A:\ EOR_K$ where $EOR_K = \{A.H(\{M\}_K).K\}_{inv(Kb)}$

where K is a symmetric key freshly generated by A, H is a one-way hash function, Kg is the public key of agent g and $inv(Kg)$ is the private key of agent g (used for signing messages).

Note that we assure that all public keys are known by all agents (including dishonest agents).

In the first message, A sends the message M encrypted by K and the evidence of origin for B (message signed by A, so decryptable by B). In this evidence, B can check his identity, learns the name of the TTP, can check that the hash code is the result of hashing the first part of the message, but cannot decrypt the last part of the evidence; this last part may be useful if any of the other sub-protocols is used.

B answers by sending the evidence of receipt for A, A checking that EOR_M is EOO_M signed by B.

In the third message, A sends the key K, permitting B to discover the message M.

Finally, B sends to A another evidence of receipt, permitting A to check that the symmetric key has been received by B.

2.2 The *Abort* Sub-protocol

The *abort* sub-protocol is executed by agent A in case he does not receive the message EOR$_M$ at step 2 of the main protocol. The purpose of this sub-protocol is to cancel the messages exchange.

1. $A \rightarrow TTP:\ \{\texttt{abort}.H(\{M\}_K).B.\{K.A\}_{Kttp}\}_{inv(Ka)}$
2. $TTP \rightarrow A:\ \begin{cases} E_{TTP} & \text{where } E_{TTP} = \{A.B.K.H(\{M\}_K)\}_{inv(Kttp)} \\ & \text{if } \texttt{resolved}(A.B.K.H(\{M\}_K)) \\ AB_{TTP} & \text{where } AB_{TTP} = \{A.B.H(\{M\}_K).\{K.A\}_{Kttp}\}_{inv(Kttp)} \\ & \text{otherwise} \end{cases}$

In this sub-protocol, A sends to the TTP an abort request, containing the `abort` label and some information about the protocol session to be aborted: the hash of the encrypted message, the name of the other agent (B), and the key used for encrypting M.

According to what happened before, the TTP has two possible answers: if this is the first problem received by the TTP for this protocol session, the TTP sends a confirmation of abortion, and stores in its database that this protocol session has been aborted; but if the TTP has already received a request for resolving this protocol session, he sends to A the information for completing his evidence of receipt by B.

2.3 The *Resolve* Sub-protocol

The role of this second sub-protocol is to permit agents A and B to finish the protocol in a fair manner, if the main protocol cannot be run until its end by some of the parties. For example, if B does not get K or if A does not get EOR_K, they can invoke the *resolve* sub-protocol.

1. $G \rightarrow TTP: \; EOR_M$
2. $TTP \rightarrow G: \begin{cases} AB_{TTP} & \text{if } \texttt{aborted}(A.B.K.H(\{M\}_K)) \\ E_{TTP} & \text{otherwise} \end{cases}$

where G stands for A or B.

A resolve request is done by sending EOR_M to the TTP. If the protocol session has already been aborted, the TTP answers by the abortion confirmation. If this is not the case, the TTP sends E_{TTP} so that the user could complete its evidence of receipt (if G is A) or of origin (if G is B). Then the TTP stores in its database that this protocol session has been resolved.

2.4 Agents' Evidences

Non-repudiation protocols require evidence of receipt (EOR) and evidence of origin (EOO). All parties have to agree that these evidences constitute a valid proof of participation in the protocol. In the case of a dispute, the parties should present their evidences to an external judge. Ideally the judge should be capable of deciding the matter by executing a verification algorithm over the evidences presented by each party.

For the CCD protocol, the evidence of receipt for A is $\{M\}_K$ and EOR_M, plus either EOR_K or E_{TTP}. The evidence of origin for B is $\{M\}_K$, EOO_M and K. At the end of the protocol execution, each agent must have all the parts that compose his evidence.

The choice of these evidences is not discussed here, see [3] for more information.

3 Analysis of the CCD Protocol

The CCD protocol was formally analysed by its authors in [3] and no attack has been found for the following scenarios: A and B honest; A honest, B dishonest; and B dishonest, A honest.

But our analysis shows that there is a serious flaw in the protocol, even when the agents act honestly. The attack occurs because one agent does not get all the required information for building its evidence when the protocol finishes by the intervention of the TTP. We describe in Sections 3.3 and 3.4 two scenarios that lead to an unfair situation for the agent playing the role A, thus contradicting the result of [3] for the same fairness property. But before presenting the attacks, we describe in the next sections the AVISPA Tool analysis method and the representation of the non-repudiation properties in the AVISPA Tool.

3.1 Analysis Method

Our analysis method is based on the technology build into the AVISPA Tool [1]: the protocol is specified in the High Level Protocol Specification Language (HLPSL) [4], translated into a state transition system called the intermediate format (IF) and fed to one of the four analysis engines available with the tool. In this work, the Attack Searcher (AtSe) engine [15] has been used. The AtSe analysis engine implements the so-called lazy intruder model [5], which greatly increases the performance of the searching process. Previously only used to analyse secrecy and authentication properties, we have extended this engine to support a subset of Linear Temporal Logics (LTL) formulae, allowing the specification and analysis of a broader spectrum of properties, including the fairness property for non-repudiation.

3.2 Description of Non-repudiation Properties

The AVISPA Tool was designed to analyse complex Internet security protocols, like the protocols described by the Internet Engineering Task Force (IETF). Even though the tool has support for the specification of arbitrarily complex properties by the use of LTL formulae, no analysis engine of the AVISPA Tool actually uses this power. Natively, properties are specified by the use of macros and only secrecy and authentication properties are supported.

In a previous work [12], we have represented non-repudiation properties as a combination of authentication properties. This representation has been applied to the Fair Zhou-Gollmann protocol [16] and has given good results, raising a problem in the protocol. But because of the implementation of the intruder strategy in the AVISPA Tool, the notion of dishonest agent could not be fully expressed (see [12] for more details). This is the reason why we have decided to use LTL formulae for describing non-repudiation properties in HLPSL, and to extend AtSe for considering this kind of formulae.

The main role of a non-repudiation protocol is to give evidences of non-repudiation to the parties involved in the protocol. To analyse this kind of protocol, one must verify which participants have their non-repudiation evidences at the end of the protocol execution. If the originator has all the parts of its non-repudiation evidence, then non-repudiation of reception is guaranteed. If the recipient has all the parts of its non-repudiation evidence, then non-repudiation of origin is guaranteed. If both parties (or none of them) have their evidences, fairness is guaranteed. In other words, to analyse non-repudiation, we need to verify if a set of terms is known by an agent at the end of the protocol execution.

To analyse non-repudiation in the AVISPA Tool, we have to find a way to express the knowledge of the agents by a predicate added in some protocol transitions, and to find a way to express the non-repudiation properties by the use of these predicates. We have then introduced the predicates `aknows` (for agent knowledge) and `iknows` (for intruder knowledge) in all the levels of the AVISPA Tool, namely in the specification language (HLPSL), in the intermediate format (IF) and in the analysis engine (AtSe). Note that `iknows` was already used in the IF and in AtSe. As with the other predicates, `aknows` and `iknows` are used in the LTL description of the properties (non-repudiation properties in our case) and to mark the protocol specification.

Definition 1 (`aknows`). *Le $\mathcal{A}$ be a set of agents playing a finite number of sessions $\mathcal{S}$ of a protocol, $\mathcal{T}$ a set of terms sent in the messages of this protocol and $\mathcal{E}$ the subset of terms $t \in \mathcal{T}$ that are part of the evidences of non-repudiation in the protocol. For an agent $a \in \mathcal{A}$, $\mathcal{E}_a$ is the set of terms $t \in \mathcal{E}$ that constitute the evidence of non-repudiation for the agent a. The predicate **aknows**(a, b, s, t) with $a, b \in \mathcal{A}$, $s \in \mathcal{S}$ and $t \in \mathcal{T}$, express that the agent a, playing with agent b in the session s, knows the term t.*

Definition 2 (Non-repudiation of origin or receipt). *If at the end of the execution of agent a in protocol session s, the predicate **aknows**(a, b, s, t) is true for all $t \in \mathcal{E}_a$, then the non-repudiation property (of origin or receipt, according to the role of a in the protocol) is satisfied. Otherwise, the property of non-repudiation for agent a is false.*

The fairness of the non-repudiation property is true only when both agents know their non-repudiation evidences, or when neither one nor the other knows his evidence. But for the properties of non-repudiation of origin and non-repudiation of receipt, the knowledge of one agent is enough to decide if the property is true or not.

With the predicates `aknows` and `iknows`, we know exactly when an agent learns a term t and thus we can automatically verify the non-repudiation properties using the knowledge of the agents. If at the end of the execution of an agent, there is no `aknows` for the non-repudiation evidences of that agent, then we have a non-repudiation of origin or non-repudiation of receipt attack.

Definition 3 (Fairness). *If at the end of the execution of agent a in session s, the predicate **aknows**(a, b, s, t) is true for all $t \in \mathcal{E}_a$, then the fairness property is true from the point of view of a. And if the fairness property is true from the point of view of the other agent, say b, the protocol session is said to be fair. The protocol is also fair if none of the agents knows all his evidences. Otherwise, the fairness property is false.*

Even if the fairness property needs data from both agents, when the predicate `aknows` is true for one agent, agent a for example, we can guarantee that the property is satisfied from the point of view of a and concentrate the analysis on the property by the point of view of agent b at the end of his execution. If one agent is dishonest or personified by the intruder, say b for example, the predicate $\texttt{aknows}(b, a, s, u)$ must be replaced by $iknows(u)$ and the agent name is written i (the intruder name). This last predicate is satisfied if the intruder knows (or can build from his knowledge) the term u.

The AtSe analysis engine has been extended to analyse properties described as LTL formulae using `aknows` and `iknows` predicates. The non-repudiation fairness for the CCD protocol is described by the following LTL formula:

$$\Box \left(\begin{pmatrix} (\ \texttt{aknows}(A, B, s, \{M\}_K) \wedge \texttt{aknows}(A, B, s, EOR_M) \wedge \\ (\texttt{aknows}(A, B, s, EOR_K) \vee \texttt{aknows}(A, TTP, s, E_{TTP}))\ ) \vee \\ (\ \texttt{iknows}(\{M\}_K) \wedge \texttt{iknows}(EOR_M)\ \wedge A = i \wedge \\ (\texttt{iknows}(EOR_K) \vee \texttt{iknows}(E_{TTP}))\ ) \end{pmatrix} \Rightarrow \begin{pmatrix} \texttt{aknows}(B, A, s, \{M\}_K) \wedge \\ \texttt{aknows}(B, A, s, EOO_M) \wedge \\ (\ \texttt{aknows}(B, A, s, K) \vee \\ \texttt{aknows}(B, TTP, s, E_{TTP})\ ) \end{pmatrix} \right)$$

Basically the property states that if A knows the EOR evidence ($\{M\}_K$, EOR_M, and EOR_K or E_{TTP}) or if the intruder, playing the role A, knows this evidence, then B must know the EOO evidence. There is a similar property for B: if B knows the EOO evidence ($\{M\}_K$, EOO_M, and K or E_{TTP}) or if the intruder knows it, then A must know the EOR evidence:

$$\Box \left(\begin{pmatrix} (\ \texttt{aknows}(B, A, s, \{M\}_K) \wedge \texttt{aknows}(B, A, s, EOO_M) \wedge \\ (\texttt{aknows}(B, A, s, K) \vee \texttt{aknows}(B, TTP, s, E_{TTP}))\ ) \vee \\ (\ \texttt{iknows}(\{M\}_K) \wedge \texttt{iknows}(EOO_M)\ \wedge B = i \wedge \\ (\texttt{iknows}(K) \vee \texttt{iknows}(E_{TTP}))\ ) \end{pmatrix} \Rightarrow \begin{pmatrix} \texttt{aknows}(A, B, s, \{M\}_K) \wedge \\ \texttt{aknows}(A, B, s, EOR_M) \wedge \\ (\ \texttt{aknows}(A, B, s, EOR_K) \vee \\ \texttt{aknows}(A, TTP, s, E_{TTP})\ ) \end{pmatrix} \right)$$

The protocol was specified in the HLPSL language and analysed with the new version of the AtSe engine. The attacks found in the analysis are described in the following sections.

3.3 Delayed Abort Request Attack

When A does not receive EOR_M from B, the *abort* sub-protocol is invoked. When B does not receive K from A, the *resolve* sub-protocol is invoked. So, if the messages EOR_M and K are not sent or delayed in the insecure channel between A and B (either because of a network problem, or intercepted by the intruder), both agents will query the TTP, A trying to abort and B trying to resolve the protocol.

The problem arises if the abort request does not reach the TTP before the resolve request. In this case, the TTP will resolve the protocol, permitting B to get all the knowledge for building the evidence of origin. Because of this previous resolve request by B, the abort request by A will not lead to the abortion of the protocol. If the TTP receives this abort request, he will send E_{TTP} to A, but as A does not (and cannot) know EOR_M, he cannot build the evidence of receipt. So, at the end of the execution, there is a fairness attack, as B can prove that A has sent M, but A cannot prove that B has received it.

The attack trace given below, automatically found by AtSe, is even more surprising, as explained hereafter. In this trace, $i(G)$ means that the intruder impersonated agent G; and for a better clarity, the detailed contents of messages have been replaced by more explicit names.

```
1. A -> i(B)   : {M}_K.EOOM
*** timeout for A ***
2. A -> i(TTP) : ABORT
3. i(A) -> B   : {M}_K.EOOM
4. B -> i(A)   : EORM
5. i(A) -> TTP : RESOLVE  (=EORM)
6. TTP -> i(A) : ETTP
*** timeout for B ***
7. B -> i(TTP) : RESOLVE
8. i(TTP) -> A : ETTP
9. i(TTP) -> B : ETTP
```

The first step is the standard one, but the intruder intercepts the message before it reaches B. Without any answer to his message, A decides to abort the protocol, message also intercepted by i (step 2). In step 3, i impersonating A forwards the message 1 to B, who answers with EOR_M (step 4). The intruder uses this last message for pretending to the TTP that A wants to resolve the protocol (step 5). As the TTP has not received the abort request of A, he answers by sending E_{TTP} (step 6). B not having any answer to his EOR_M message, he decides to ask the TTP for resolving the protocol (step 7). Then the intruder sends the TTP resolve answer to A and B (steps 8 and 9).

The originality of this attack trace is that, at the end:

– A will guess (according to the answer received to his abort request) that the protocol has been resolved by B, so he will assume that B knows M and can build the proof that A has sent it; but A cannot prove this;
– B has resolved the protocol and has received from the TTP the information for getting M and building the proof that A has sent M; but he does not know that A does not have his proof;
– the TTP will think that A has asked for the protocol to be resolved, followed by B; so for him, both A and B can build their evidences.

So, this trace shows that the CCD protocol is not fair, even if both agents A and B are honest. The attack is due to a malicious intruder, and the TTP is of no help for detecting the problem.

3.4 Dishonest Agent Attack

A variant of the previous attack has also been discovered by AtSe. It happens when agent A plays the protocol with a dishonest agent B (called the intruder and names i). As soon as i has received the first message from A, he builds EOR_M and sends it to the TTP as resolve request. When A decides to abort the protocol, this is too late: the protocol has already been resolved, the intruder can get M and build the proof that A has sent M, and A cannot build the evidence of receipt.

```
1. A -> i   : {M}_K.EOOM
2. i -> TTP : RESOLVE
3. TTP -> i : ETTP
*** timeout for A ***
4. A -> TTP : ABORT
5. TTP -> A : ETTP
```

4 Correction of the CCD Protocol

In this section, we first discuss the role of the trusted third party for trying to solve the problems raised by the attacks found. Then we describe a correction of the *abort* sub-protocol and report the new analyses done, in which no attack has been found.

4.1 About the TTP Role

Both attacks described in the previous section come from the same flaw: the TTP does not give EOR_M to agent A when the protocol is already resolved and A tries to abort it. However, the TTP has received EOR_M in the resolve request, so one can argue that A only needs to know E_{TTP} to prove that B knows the message M: A knowing E_{TTP} means that TTP knows EOR_M, and consequently A could know EOR_M by asking it to the TTP, in case of a dispute.

From B's side, if B resolves the protocol and gets the message E_{TTP}, this means that B knows EOR_M, and according to the protocol, owning EOR_M means owning EOO_M and M_K. If the TTP stores EOR_M in its database for every resolved transaction, A could try to prove that B knows M by requesting to the TTP a proof that EOR_M is known by B.

If we consider this situation acceptable, and if we prove that A knowing E_{TTP} implies B also knowing E_{TTP} and M_K, we can say that the protocol is fair even when A only receives E_{TTP} as evidence of receipt.

But this situation is not acceptable, first because accepting E_{TTP} as an evidence of receipt puts extra importance on the TTP. The evidences should be strong enough to prove participation in the protocol without the need of using TTP's knowledge as part of the proof. Second, the TTP would need to store all EOR_M messages for all resolved sessions of the protocol. And last, without EOR_M we cannot prove that B has agreed on the use of the agent TTP as the trusted third party: there is no message signed by B that contains the name of the TTP. So E_{TTP} cannot be a proof of receipt without EOR_M.

This is why we propose some changes to correct this flaw in the protocol.

4.2 Correction of the *abort* Sub-protocol

To correct the protocol, we need to change the *abort* sub-protocol to provide the complete EOR evidence to A, no matter the sequence of abort and resolve requests in the session of the protocol. Below we present the new version of the *abort* sub-protocol.

1. $A \to TTP: \{\texttt{abort}.H(\{M\}_K).B.\{K.A\}_{Kttp}\}_{inv(Ka)}$
2. $TTP \to A: \begin{cases} E_{TTP}.EOR_M & \text{if } \texttt{resolved}(A.B.K.H(\{M\}_K)) \\ AB_{TTP} & \text{otherwise} \end{cases}$

Messages E_{TTP}, EOR_M and AB_{TTP} are the same as in the original protocol. The only change is the addition of EOR_M message in the TTP's answer to A when the sub-protocol is invoked and the TTP has already resolved the session (and stored EOR_M together with the `resolved` predicate in its database).

We have specified and analysed the corrected protocol. An extended number of scenarios has been checked, compared to the original work of Cederquist et al. [3], including two-sessions scenarios where the sessions are run in parallel.

One-session scenarios. We have analysed the common one-session scenarios: A and B honest, A honest and B dishonest, A dishonest and B honest. In our analysis approach, the intruder impersonates the dishonest agents. For all three scenarios the fairness property could not be falsified.

Two-sessions scenarios. We have also analysed some critical two-sessions scenarios: A and B honest in parallel with A honest and B dishonest; A and B honest in parallel with A dishonest and B honest; A honest and B dishonest in parallel with A dishonest and B honest. When running sessions in parallel, the intruder has an improved knowledge and he can try, for example, to use knowledge/messages from one session in the other session. Again, for those scenarios AtSe has found no fairness attack.

5 Conclusion

Non-repudiation protocols have an important role in many areas where secured transactions with proofs of participation are necessary. The evidences of origin and receipt of a message are the elements that the parties should have at the end of the communication. The CCD protocol is a recent non-repudiation protocol that avoids the use of session labels and distinguishes itself by the use of an optimistic approach, the Trusted Third Party being used only in case of a problem in the execution of the main protocol.

The fairness of a non-repudiation protocol is a property difficult to analyse and there are very few tools that can handle the automatic analysis of this property. The contribution of this work is twofold. First we have extended the AVISPA Tool and one of its analysis engines, AtSe, to implement our analysis method for the non-repudiation properties. Our method is based on the knowledge of agents and can be used to automatically analyse non-repudiation protocols as well as contract signing protocols [14]. Second, with this method, we have specified and analysed the CDD protocol and a serious flaw has been found. We have proposed a correction that has been further analysed by additional scenarios and no attack has been found.

Our representation of the non-repudiation properties has also been applied successfully to the Fair Zhou-Gollmann protocol [12]. We have tested other specifications of the CCD protocol, for example with secure communication channels between agents and the TTP, and for the original definition for the *abort* sub-protocol: no attack has been found; but using such channels is not considered as acceptable, because it requires too much work for the TTP.

The AVISPA Tool has proved its efficiency for analysing secrecy and authentication properties of protocols. We have extended it to handle non-repudiation properties, but by this extension, adding `aknows` and `iknows` predicates and using LTL formulae as goal, we have open a highway to the specification of many other properties, without any more change in the specification languages and the analysis engines. And for the analysis of the CCD protocol, the use of LTL formulae did not have any impact on the speed of AtSe for finding attacks (or for not finding attacks concerning the fixed version of the protocol).

References

1. Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, UK, 2005. Springer.
2. Giampaolo Bella and Lawrence C. Paulson. Mechanical Proofs about a Non-repudiation Protocol. In Richard J. Boulton and Paul B. Jackson, editors, *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001*, volume 2152 of *Lecture Notes in Computer Science*, pages 91–104, Edinburgh, Scotland, UK, 2001. Springer.
3. Jan Cederquist, Ricardo Corin, and Muhammad Torabi Dashti. On the Quest for Impartiality: Design and Analysis of a Fair Non-repudiation Protocol. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors, *Information and Communications Security, 7th International Conference, ICICS 2005*, volume 3783 of *Lecture Notes in Computer Science*, pages 27–39, Beijing, China, 2005. Springer.
4. Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Jacopo Mantovani, Sebastian Mödersheim, and Laurent Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Automated Software Engineering. Proceedings of the Workshop on Specification and Automated Processing of Security Requirements, SAPS'04*, pages 193–205, Austria, September 2004. Austrian Computer Society.
5. Yannick Chevalier and Laurent Vigneron. A Tool for Lazy Verification of Security Protocols. In *16th IEEE International Conference on Automated Software Engineering (ASE 2001)*, pages 373–376, San Diego, CA, USA, 2001. IEEE Computer Society.

6. Sigrid Gürgens, Carsten Rudolph, and Holger Vogt. On the Security of Fair Non-repudiation Protocols. In Colin Boyd and Wenbo Mao, editors, *Information Security, 6th International Conference, ISC 2003*, volume 2851 of *Lecture Notes in Computer Science*, pages 193–207, Bristol, UK, 2003. Springer.
7. Steve Kremer, Olivier Markowitch, and Jianying Zhou. An Intensive Survey of Fair Non-repudiation Protocols. *Computer Communications Journal*, 25(17): 1606–1621, 2002.
8. Olivier Markowitch and Steve Kremer. An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party. In George I. Davida and Yair Frankel, editors, *Information Security, 4th International Conference, ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378, Malaga, Spain, 2001. Springer.
9. Olivier Markowitch and Yves Roggeman. Probabilistic Non-Repudiation without Trusted Third Party. In *Second Workshop on Security in Communication Networks'99*, Amalfi, Italy, 1999.
10. Henning Pagnia and Felix C. Gärtner. On the Impossibility of Fair Exchange without a Trusted Third Party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Darmstadt, Germany, 1999.
11. Peter Ryan, Michael Goldsmith, Gavin Lowe, Bill Roscoe, and Steve Schneider. *Modelling & Analysis of Security Protocols*. Addison Wesley, 2000.
12. Judson Santiago and Laurent Vigneron. Automatically Analysing Non-repudiation with Authentication. In *Proceedings of 3rd Taiwanese-French Conference on Information Technology (TFIT)*, pages 541–554, Nancy, France, March 2006.
13. Steve Schneider. Formal Analysis of a Non-Repudiation Protocol. In *Proceedings of The 11th Computer Security Foundations Workshop*, pages 54–65. IEEE Computer Society Press, 1998.
14. Vitaly Shmatikov and John C. Mitchell. Analysis of Abuse-Free Contract Signing. In Yair Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 174–191, Anguilla, British West Indies, 2000. Springer.
15. Mathieu Turuani. The CL-Atse Protocol Analyser. In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA 2006*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286, Seattle, WA, USA, 2006. Springer.
16. Jianying Zhou and Dieter Gollmann. A Fair Non-repudiation Protocol. In *1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, USA, 1996. IEEE Computer Society.
17. Jianying Zhou and Dieter Gollmann. Towards verification of non-repudiation protocols. In *Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380, Canberra, Australia, September 1998.

Secure Remote User Authentication Scheme Using Bilinear Pairings

Eun-Jun Yoon[1], Wan-Soo Lee[2], and Kee-Young Yoo[2,⋆]

[1] Faculty of Computer Information, Daegu Polytechnic College,
395 San 130 Manchon-Dong, SooSung-Gu, Daegu 706-711, South Korea
ejyoon@tpic.ac.kr
[2] Department of Computer Engineering, Kyungpook National University,
1370 Sankyuk-Dong, Buk-Gu, Daegu 702-701, South Korea
Tel.: +82-53-950-5553; Fax: +82-53-957-4846
complete2@infosec.knu.ac.kr, yook@knu.ac.kr

Abstract. In 2006, Das et al. proposed a remote user authentication scheme using the properties of bilinear pairings. The current paper, however, demonstrates that Das et al.'s scheme is still vulnerable to an impersonation attack and an off-line password guessing attack. Furthermore, we present an improved authentication scheme based on bilinear computational Diffie-Hellman problem and one-way hash function to the schemes, in order to isolate such problems.

Keywords: Authentication, Password, Key agreement, Cryptanalysis, Smart card, Bilinear pairings.

1 Introduction

Remote user authentication is an important part of security, along with confidentiality and integrity, for systems that allow remote access over untrustworthy networks, like the Internet. As such, a remote password authentication scheme authenticates the legitimacy of users over an insecure channel, where the password is often regarded as a secret shared between the remote system and the user. With knowledge of the password, the user can use it to create and send a valid login message to a remote system in order to gain access. Meanwhile, the remote system also uses the shared password to check the validity of the login message and to authenticate the user.

ISO 10202 standards have been established for the security of financial transaction systems that use integrated circuit cards (IC cards or smart cards). The smart card originates from the IC memory card which has been in the industry for about 10 years [1][2]. The main characteristics of a smart card are its small size and low-power consumption. In general, a smart card contains a microprocessor which can quickly manipulate logical and mathematical operations, RAM, which is used as a data or instruction buffer, and ROM which stores the

⋆ Corresponding author.

user's secret key and the necessary public parameters and algorithmic descriptions of the executing programs. The merits of a smart card regarding password authentication are its simplicity and its efficiency in terms of the log-in and authentication processes.

In 2000, Joux [3] discovered the bilinear computational Diffie-Hellman problem of the groups over elliptic curves. This hard problem can be considered as a new security assumption to develop cryptosystems. Since then, several variant security schemes have been presented [4][5][6][7]. Bilinear pairings are an effective method to reduce the complexity of the discrete log problem in a finite field and provides a good setting for the bilinear computational Diffie-Hellman problem.

In 2006, Das et al. [8] proposed a remote user authentication scheme using the properties of bilinear pairings that can prohibit the scenario of many logged in users with the same login-ID, and provide a flexible password change option to the registered users without any assistance from the remote system. The current paper, however, demonstrates that Das et al.'s scheme is still vulnerable to an impersonation attack [9], where an attacker easily masquerade as another legal users in order to access the resources of a remote system, and an off-line password guessing attack [10], where an attacker can easily guess a legal users's password and can impersonate an legal users. Furthermore, we present an improved authentication scheme based on bilinear computational Diffie-Hellman problem [3] and one-way hash function [9] to the schemes, in order to isolate such problems. As a result, the proposed scheme is more secure than Das et al.'s scheme. Also, it provides mutual authentication between the user and remote system and it has the same advantages of other schemes. In addition, the proposed scheme does not require time synchronization or delay-time limitations between the user and remote system, unlike Das et al.'s scheme.

The remainder of this paper is organized as follows: In the next section, we give some preliminaries of bilinear pairings. Section 3 briefly reviews Das et al.'s scheme and then Section 4 demonstrates the security weakness of Das et al.'s scheme. The proposed authentication scheme is presented in Section 5, while Sections 6 discusses the security of the proposed protocol. The conclusion is given in Section 7.

2 Preliminaries

This section summarizes the underlying primitives used throughout this paper. This primitive include modified Weil pairing, bilinear computational Deffie-Hellman assumption, symmetric encryption scheme, one-way hash function and map-to-point function [3][7][8].

2.1 Bilinear Pairings

Suppose G_1 is an additive cyclic group generated by P, whose order is a prime q, and G_2 is a multiplicative cyclic group of the same order. A map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is called a bilinear mapping if it satisfies the following properties:

1. Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$, for all $P, Q \in G_1$ and all $a, b \in Z_q^*$.
2. Non-degenerate: there exists $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$.
3. Computable: there is an efficient algorithm to compute $\hat{e}(P, Q)$ for all P, $Q \in G_1$.

We note that G_1 is the group of points on an elliptic curve and G_2 is a multiplicative subgroup of a finite field. Typically, the mapping $\hat{e}$ will be derived from either the Weil or the Tate pairing on an elliptic curve over a finite field.

2.2 Mathematical Problems

Definition 1. *Discrete Logarithm Problem (DLP): Given $Q, R \in G_1$, find an integer $x \in Z_q^*$ such that $R = xQ$.*

The MOV and FR reductions: Menezes et al. [11] and Frey and Ruck [12] show a reduction from the DLP in G_1 to the DLP in G_2. The reduction is: Given an instance $Q, R \in G_1$, where Q is a point of order q, find $x \in Z_q^*$, such that $R = xQ$. Let T be an element of G_1 such that $g = \hat{e}(T, Q)$ has order q, and let $h = \hat{e}(T, R)$. Using bilinear property of $\hat{e}$, we have $\hat{e}(T, R) = \hat{e}(T, Q)^x$. Thus, DLP in G_1 is no harder than the DLP in G_2.

Definition 2. *Bilinear Computational Diffie-Hellman Problem (BCDHP): Given (P, aP, bP) for $a, b \in Z_q^*$, compute abP.*

The advantage of any probabilistic polynomial-time algorithm $\mathcal{A}$ in solving the BCDHP in G_1, is defined as $Adv_{\mathcal{A},G_1}^{CDH} = Prob[\mathcal{A}(P, aP, bP, abP) = 1 : a, b \in Z_q^*]$. For every probabilistic algorithm A, $Adv_{\mathcal{A},G_1}^{CDH}$ is negligible.

3 Review of Das et al.'s Scheme

This section briefly reviews Das et al.'s authentication scheme [8]. Das et al.'s scheme consists of mainly three phases: Setup, registration, and authentication phase. Figure 1 shows Das st al.'s authentication scheme. The scheme works as follows:

3.1 Setup Phase

Let G_1 is an additive cyclic group of order prime q, and G_2 is a multiplicative cyclic group of the same order. Let P is a generator of G_1, $\hat{e} : G_1 \times G_1 \in G_2$ is a bilinear mapping and $H : \{0,1\}^* \rightarrow G_1$ is a cryptographic hash function. The remote system RS selects a secret key s and computes the public-key as $Pub_{RS} = sP$. Then, the RS publishes the system parameters $< G_1, G_2, \hat{e}, q, P, Pub_{RS}, H(\cdot) >$ and keeps s secret.

3.2 Registration Phase

This phase is executed by the following steps when a new user wants to register with the RS:

R1. Suppose a new user U_i wants to register with the RS, then U_i submits his identity ID_i and password PW_i to the RS.
R2. On receiving the registration request, the RS computes $Reg_{ID_i} = s \cdot H(ID_i) + H(PW_i)$.
R3. The RS personalizes a smart card with the parameters ID_i, Reg_{ID_i}, $H(\cdot)$ and sends the smart card to U_i over a secure channel.

3.3 Authentication Phase

This phase is executed every time whenever a user logs into the RS. The phase is further divided into the login and verification phases. In the login phase, user sends a login request to the RS. The login request comprises with a dynamic coupon, called DID, which is dependent on the user's ID, password and RS's secret key. The RS allows the user to access the system only after successful verification of the login request.

Login Phase: The user U_i inserts the smart card in a terminal and keys ID_i and PW_i. If ID_i is identical to the one that is stored in the smart card, the smart card performs the following operations:

L1. Computes $DID_i = T \cdot Reg_{ID_i}$, where T is the user system's timestamp.
L2. Computes $V_i = T \cdot H(PW_i)$.
L3. Sends the login request $\{ID_i, DID_i, V_i, T\}$ to the RS over a public channel.

Verification Phase: Let the RS receives the login message $\{ID_i, DID_i, V_i, T\}$ at time T^* $(\geq T)$. The RS performs the following operations to verify the login request:

V1. Verifies the validity of the time interval between T^* and T. If $(T^* - T) \leq \Delta T$, the RS proceeds to the Step (V2), where ΔT denotes the expected valid time interval for transmission delay. Otherwise, rejects the login request. We note that at the time of registration, the user and the RS have agreed on the accepted value of the transmission delay ΔT.
V2. Checks whether $\hat{e}(DID_i - V_i, P) = \hat{e}(H(ID_i), Pub_{RS})^T$. If it holds, the RS accepts the login request; otherwise, rejects it.

3.4 Password Change Phase

When a user U_i wants to change his password, U_i can change his password without taking any assistance from the RS by invoking this phase. Das et al.'s password change phase works as follows:

Shared Information: $G_1, G_2, \hat{e}, q, P, Pub_{RS}, H(\cdot)$.
Information held by User U_i: ID_i, PW_i, Smart card$(ID_i, Reg_{ID_i}, H(\cdot))$.
Information held by Remote System RS: s.

User U_i		**Remote System**
Registration Phase:		
Select ID_i, PW_i	$\xrightarrow{ID_i, PW_i}$	
		$Reg_{ID_i} \leftarrow s \cdot H(ID_i) + H(PW_i)$
		Store $ID_i, Reg_{ID_i}, H(\cdot)$ in Smart Card
	$\xleftarrow[\text{(Secure Channel)}]{\text{Smart Card}}$	
Login and Verification Phase:		
Input ID_i,PW_i		
Pick up T		
$DID_i \leftarrow T \cdot Reg_{ID_i}$		
$V_i \leftarrow T \cdot H(PW_i)$	$\xrightarrow{\{ID_i, DID_i, V_i, T\}}$	Check $(T^* - T) \leq \Delta T$
		Check $\hat{e}(DID_i - V_i, P) \overset{?}{=} \hat{e}(H(ID_i), Pub_{RS})^T$

Fig. 1. Das et al.'s Authentication Scheme

P1. U_i attaches the smart card to a terminal and keys ID_i and PW_i. If ID_i is identical to the one that is stored in the smart card, proceeds to the Step (P2); otherwise, terminates the operation.

P2. U_i submits a new password PW_i^*.

P3. The smart card computes $Reg_{ID_i}^* = Reg_{ID_i} - H(PW_i) + H(PW_i^*) = s \cdot H(ID_i) + H(PW_i^*)$.

P4. The password has been changed now with the new password PW_i^* and the smart card replaced the previously stored Reg_{ID_i} value by $Reg_{ID_i}^*$ value.

4 Cryptanalysis of Das et al.'s Scheme

This section demonstrates that Das et al.'s authentication scheme is vulnerable to some attacks.

4.1 Impersonation Attack

This subsection demonstrates that Das et al.'s scheme is vulnerable to an impersonation attack, where an attacker can easily impersonate other legal users to access the resources at a remote system. Suppose that an attacker E has eavesdropped a valid message (ID_i, DID_i, V_i, T) from an open network. It is easy to obtain the information since it is exposed over an open network. Then, in the Login Phase, an impersonation attack proceeds as follows:

(1) E chooses a timestamp T' and computes $r = T'/T$, where T' is E's the current date and time for succeeding with Step (V2) of the Authentication Phase.
(2) E computes $DID'_i = r \cdot DID_i$ and $V'_i = r \cdot V_i$.
(3) E sends a forged message (ID_i, DID'_i, V'_i, T') to RS.
(4) It is easy to check whether RS will accept this forged message, as $\hat{e}(DID'_i - V'_i, P) \overset{?}{=} \hat{e}(H(ID_i), Pub_{RS})^{T'}$. Its correctness easy to see that the Verification Step (V2) of E's forged login request is verified by the following:

$$\begin{aligned}
\hat{e}(DID'_i - V'_i, P) &= \hat{e}(r \cdot DID_i - r \cdot V_i, P) \\
&= \hat{e}(r \cdot T \cdot Reg_{ID_i} - r \cdot T \cdot H(PW_i), P) \\
&= \hat{e}(r \cdot T \cdot (s \cdot H(ID_i) + H(PW_i)) - r \cdot T \cdot H(PW_i), P) \\
&= \hat{e}(r \cdot T \cdot s \cdot H(ID_i), P) \\
&= \hat{e}(s \cdot H(ID_i), P)^{r \cdot T} \\
&= \hat{e}(H(ID_i), sP)^{T'} \\
&= \hat{e}(H(ID_i), Pub_{RS})^{T'}
\end{aligned}$$

(5) Finally, RS will accept the attacker's login request, making Das et al.'s scheme insecure.

4.2 Off-Line Password Guessing Attack

In the login phase of Das et al.'s scheme, suppose that an attacker E has eavesdropped a valid message (ID_i, DID_i, V_i, T) from an open network. Then, in order to obtain the password PW_i of user U_i, the off-line password guessing attack proceeds as follows:

(1) E makes a guess at the secret password PW'_i.
(2) E computes $T \cdot H(PW'_i)$, where T is intercepted U_i's current timestamp.
(3) E checks if $V_i = T \cdot H(PW'_i)$.
(4) If the computed value is the same as V_i, then E guesses the legitimate user U_i's password PW_i. Otherwise, E repeatedly performs Steps (1), (2) and (3) until $V_i = T \cdot H(PW'_i)$.

If a user loses his smart card and it is found out by an attacker or an attacker steals a user's smart card, then the attacker can easily impersonate the legitimate user U_i by using the guessed password PW'_i in the Login Phase. Furthermore, if some users employ the same password for multiple accounts, those will be compromised as well. As a result, Das et al.'s scheme is vulnerable to an off-line password guessing attack.

5 Proposed Scheme

This section proposes an improvement of Das et al.'s scheme so that they can withstand the above mentioned attacks. In addition, the proposed scheme provides mutual authentication between the user and a remote system and does not require time synchronization or a delay-time limitations between the user and the remote system. In order to prevent the problems of clock synchronization or a delay-time limitations, the proposed scheme adopts a nonce-based protocol [13] instead of a timestamp-based protocol. The security of the proposed scheme is based on Discrete Logarithm Problem (DLP), Bilinear Computational Diffie-Hellman problem (BCDHP) (Definitions 1, 2 in Section Preliminaries) and one-way hash function, and consists of setup, registration, and authentication phases. Figure 2 shows the proposed authentication scheme. The scheme works as follows:

5.1 Setup Phase

Let G_1 is an additive cyclic group of order prime q, and G_2 is a multiplicative cyclic group of the same order. Let P is a generator of G_1, $\hat{e} : G_1 \times G_1 \in G_2$ is a bilinear mapping, $H : \{0,1\}^* \rightarrow G_1$ is a cryptographic hash function and $F(\cdot)$ is a collision resistant one-way hash function with an output size of 512 bits, e.g. SHA-512 [9]. The remote system RS selects a secret key s. Then, the RS publishes the system parameters $< G_1, G_2, \hat{e}, q, P, H(\cdot), F(\cdot) >$ and keeps s secret.

5.2 Registration Phase

This phase is executed by the following steps when a new user wants to register with the RS:

R1. Suppose a new user U_i wants to register with the RS, then U_i selects his identity ID_i, password PW_i and random number N freely.
R2. U_i computes $F(PW_i|N)$, where $|$ is a concatenation operation, and then submits ID_i and $F(PW_i|N)$ to the RS.
R3. On receiving the registration request, the RS computes $U = H(ID_i, ID_s)$, $K_i = s \cdot U$, $VK_i = F(K_i)$ and $Reg_{ID_i} = K_i + H(F(PW_i|N))$, where ID_s is the RS's identity.
R4. The RS personalizes a smart card with the parameters U, VK_i, Reg_{ID_i}, $H(\cdot)$, $F(\cdot)$ and sends the smart card to U_i over a secure channel.
R5. U_i enters N into his smart card.

5.3 Authentication Phase

This phase is executed every time whenever a user logs into the RS. The phase is further divided into the login and session key agreement phases.

Login Phase: If the user U_i wants to login, U_i inserts the smart card in a terminal and keys ID_i and PW_i. Then, the smart card performs the following operations:

L1. Extracts K_i from the smart card by computing $Reg_{ID_i} - H(F(PW_i|N))$.
L2. Computes hash value $F(K_i)$ and verifies it with stored VK_i. If it holds, the card performs next Step. Otherwise, the card rejects U_i's login request. This verification process performs only three times that can withstand password guessing attack by using stolen or lost smart card.
L3. Chooses a fresh random value $a \in Z_q^*$, and computes $C_1 = aP$.
L4. Sends a login request message $\{ID_i, C_1\}$ to RS.

Session Key Agreement Phase: Upon receiving the authentication request message $\{ID_i, C_1\}$, the remote system and smart card execute the following steps for mutual authentication and session key agreement between the user U_i and the remote system.

K1. The system verifies the format of ID_i. If the format is incorrect, the system rejects the login request. Otherwise, the system computes $U = H(ID_i, ID_s)$ and $K_i^* = s \cdot U$. Then, the system chooses a fresh random value $b \in Z_q^*$, and computes $C_2 = bP$, $sk = \hat{e}(C_1, bU) = \hat{e}(aP, bU) = \hat{e}(P, U)^{ab}$ and $C_3 = F(ID_i, K_i^*, sk, C_1)$. The system sends back the message $\{C_2, C_3\}$.
K2. Upon receiving the message $\{C_2, C_3\}$, the smart card computes $sk^* = \hat{e}(C_2, aU) = \hat{e}(bP, aU) = \hat{e}(P, U)^{ab}$ and $C_3^* = F(ID_i, K_i, sk^*, C_1)$. Then, the smart card compares C_3 and C_3^*. If they are equal, the user U_i believes that the responding part is the real system, otherwise the user U_i interrupts the connection. Finally, the smart card computes $C_4 = F(ID_i, K_i, sk^*, C_2)$ and sends this authentication token to the system for mutual authentication and session key agreement.
K3. Upon receiving the message $\{C_4\}$, the system computes $C_4^* = F(ID_i, K_i^*, sk, C_2)$ and compares C_4 and C_4^*. If they are equal, the system can ensure that the user U_i is legal.

After mutual authentication and session key agreement between the user and the remote system, sk and sk^* are used as a session key, respectively.

5.4 Password Change Phase

This phase is invoked whenever a user U_i wants to change his password. By invoking this phase, U_i can easily change his password without taking any assistance from the RS. The phase works as follows:

P1. U_i attaches the smart card to a terminal and keys ID_i and PW_i.
P2. The smart card computes $K_i = Reg_{ID_i} - H(F(PW_i|N))$.
P3. The smart card computes hash value $F(K_i)$ and verifies it with stored VK_i. If it holds, the smart card proceeds to the Step (P4); otherwise, terminates the operation. This verification process performs only three times that can withstand password guessing attack by using stolen or lost smart card.
P4. U_i submits a new password PW_i^*.
P5. The smart card computes $Reg_{ID_i}^* = K_i + H(F(PW_i^*|N))$.
P6. The password has been changed now with the new password PW_i^* and the smart card replaced the previously stored Reg_{ID_i} value by $Reg_{ID_i}^*$ value.

Shared Information: $G_1, G_2, \hat{e}, q, P, H(\cdot),\ F(\cdot)$.
Information held by User U_i: ID_i, PW_i, Smart card$(U, VK_i, Reg_{ID_i}, H(\cdot), F(\cdot))$.
Information held by Remote System RS: s.

User U_i		**Remote System**
Registration Phase:		
Select ID_i, PW_i, N	$\xrightarrow{ID_i, F(PW_i \vert N)}$	$U \leftarrow H(ID_i, ID_s)$
		$K_i \leftarrow s \cdot U$
		$VK_i \leftarrow F(K_i)$
		$Reg_{ID_i} \leftarrow K_i + H(F(PW_i \vert N))$
		Store $U, VK_i, Reg_{ID_i}, H(\cdot), F(\cdot)$ in Smart Card
Enter N into Smart Card	$\xleftarrow[\text{(Secure Channel)}]{\text{Smart Card}}$	
Login and Session Key Agreement Phase:		
Input ID_i,PW_i		
$K_i \leftarrow Reg_{ID_i} - H(F(PW_i \vert N))$		
Verify $VK_i \overset{?}{=} F(K_i)$		
Choose random $a \in Z_q^*$		
$C_1 \leftarrow aP$	$\xrightarrow{\{ID_i, C_1\}}$	Verify ID_i
		$U \leftarrow H(ID_i, ID_s)$
		$K_i^* \leftarrow s \cdot U$
		Choose random $b \in Z_q^*$
		$C_2 \leftarrow bP$
		$sk \leftarrow \hat{e}(C_1, bU) \leftarrow \hat{e}(P, U)^{ab}$
$sk^* \leftarrow \hat{e}(C_2,\ aU) \leftarrow \hat{e}(P, U)^{ab}$	$\xleftarrow{\{C_2, C_3\}}$	$C_3 \leftarrow F(ID_i, K_i^*, sk, C_1)$
$C_3^* \leftarrow F(ID_i, K_i, sk^*, C_1)$		
Verify $C_3 \overset{?}{=} C_3^*$		
$C_4 \leftarrow F(ID_i, K_i, sk^*, C_2)$	$\xrightarrow{\{C_4\}}$	$C_4^* \leftarrow F(ID_i, K_i^*, sk,\ C_2)$
		Verify $C_4 \overset{?}{=} C_4^*$

Session Key: $sk = sk^* = \hat{e}(P, U)^{ab}$

Fig. 2. Proposed Authentication Scheme

6 Security Analysis

This section provides the proof of correctness of the proposed scheme. Here, nine security properties: passive attack, active attack, guessing attack, insider attack, known-key attack, secure password change, fast wrong password detection, mutual authentication and perfect forward secrecy, would be considered for the proposed scheme [9].

(1) The proposed scheme can resist a passive attack. If an attacker, called E, who eavesdrops on a successful proposed scheme run can make a guess at the session key by using only information obtainable over a network and a guessed value of the remote system's secret key s, E could break a Bilinear Computational Diffie-Hellman Problem (BCDHP) (Definition 2 in Section Preliminaries). The reason will be clear. Such a problem can be reduced to the computing of a keying material $\hat{e}(P, U)^{ab}$ from the value C_1 and C_2 in

the scheme. Thus, we claim that it is as difficult as to break the BCDHP. Without the ability to compute the keying material $\hat{e}(P, U)^{ab}$, the messages C_3 and C_4 do not leak any information to the passive attacker. Since the user U_i and the remote system do not leak any information either, the proposed scheme can resist a passive attack.

(2) The proposed scheme can resist an active attack. Active attacks can take many different forms, depending on what information is available to the attacker. An attacker who knows the remote system's secret key s can easily pretend to be U_i and communicate with the system. Similarly, an attacker with s can masquerade as the system when U_i tries to contact him. A man-in-the middle attack, which requires an attacker to fool both sides of a legitimate conversation, cannot be carried out by an attacker who does not know the system's secret key s. For example, suppose that attacker E wants to fool the system into thinking he is talking to U_i. First, E can compute $C_1' = eP$, where e is a fresh random value, and send it to the system. Then, the system will compute $sk = \hat{e}(C_1, bU) = \hat{e}(P, U)^{ae}$, $C_2 = bP$ and $C_3 = F(ID_i, K_i^*, sk, C_1')$, and send C_2 and C_3 to E. When E receives C_2 and C_3 from the remote system, E has to make $C_4' = F(ID_i, K_i', sk', C_2)$ and send it to the system. Since the problem is combined with the BCDHP and a secure one-way hash function, in order to compute valid C_4', E cannot guess sk' or K_i' from C_3. Thus, the proposed scheme can withstand the man-in-the-middle attack.

(3) The proposed scheme can resist guessing attack. Assume a user loses his smart card and it is found by an attacker or an attacker steals a user's smart card. The attacker, however, cannot impersonate a legitimate user U_i by using the smart card because no one can reveal the PW_i from value Reg_{ID_i} in the smart card without knowing the system's secret key s. Since the smart card verifies computed value $F(K_i)$ with stored VK_i, an attacker can perform a password guessing attack by using stolen or lost smart card. However, in the proposed scheme, this verification process performs only three times that can withstand the attack. Therefore, no one can get a legitimate user U_i's password PW_i. Even if an attacker has $K_i = s \cdot H(ID_i)$, it is extremely hard for any attacker to derive s from $K_i = s \cdot H(ID_i)$ because of Discrete Logarithm Problem (DLP) (Definition 1 in Section Preliminaries). Therefore, the proposed scheme can withstand the guessing attack.

(4) The proposed scheme can resist insider attack. In many scenarios, the user uses a common password to access several systems for his convenience. If the user login request is password-based and the RS maintains password or verifier table for login request verification, an insider of RS could impersonate user's login by stealing password and gets access of the other systems. In the registration phase of Das et al.'s scheme, user U_i's password PW_i will be revealed to remote server RS after Step (R2). If U_i uses PW_i to access several servers for his convenience, the insider of RS can impersonate U_i to access other servers. In the proposed scheme, since U_i registers to RS by presenting $ID_i, F(PW_i|N)$ instead of ID_i, PW_i, the insider of RS cannot

directly obtain PW_i without knowing of random nonce N. Therefore, the proposed scheme can withstand the insider attack.

(5) The proposed scheme can resist the known-key attack. Known-key security means that each run of a key agreement protocol between two entities U_i and a remote system should produce unique secret keys; such keys are called session keys. If the session key sk is revealed to a passive attacker E, E does not learn any new information from combining sk with publicly-visible information. This is true because the messages C_3 or C_4 do not leak any information to the attacker. We have already established that E cannot make meaningful guesses at the session key sk from the guessed passwords, and there does not appear to be an easy way for E to carry out an off-line password guessing attack. It means that the attacker, having already obtained some past session keys, cannot compromise current or future session keys. Thus, it can resist the known-key attack.

(6) The proposed scheme provides secure password change In Das et al.'s scheme, when a smart card is stolen, an unauthorized user can easily change a new password for the card in password-change phase. First, an unauthorized user inserts U_i's smart card into the smart card reader of a terminal, enters the ID_i and PW_e, where PW_e is the unauthorized user's arbitrary password, and requests a change of passwords. Since ID_i is public value and the entered ID_i is identical to the one that is stored in the smart card, the smart card will proceed to the Step (P2) of password change phase. Next, the unauthorized user enters an arbitrary new password PW_e^* and then the smart card computes $Reg_{ID_i}^* = Reg_{ID_i} - H(PW_e) + H(PW_e^*)$, which yields $s \cdot H(ID_i) + H(PW_i) - H(PW_e) + H(PW_e^*)$, and then replaces he previously stored Reg_{ID_i} with $Reg_{ID_i}^*$ without any checking. If a malicious user stole user U_i's smart card for a short time and change an arbitrary new password as above described, then the legal user U_i's succeeding login requests will be denied unless he re-registers with the remote server again because $\hat{e}(DID_i - V_i, P) \neq \hat{e}(H(ID_i), Pub_{RS})^T$ in the verification phase. So considered, Das et al.'s password change phase is insecure. However, the proposed scheme provides secure password change. Because the smart card can verify K_i using the stored $F(K_i)$ in Step (P3) of the password change phase, when the smart card was stolen, unauthorized users cannot change the password of the card without knowing the U_i's password PW_i. Therefore, the proposed scheme provides secure password change.

(7) The proposed scheme provides fast wrong password detection In Das et al.'s scheme, if user U_i input a wrong password by mistake, this wrong password will be detected by the remote system in the authentication phase. Therefore, Das et al.'s scheme is slow to detect the user's wrong password. In contrast to Das et al.'s scheme, in the proposed scheme, if user U_i inputs the wrong password by mistake, this wrong password will be quickly detected by a smart card since the smart card can verify $F(K_i) = VK_i$ using the stored VK_i in Step (L2) of the login phase. Therefore, the proposed scheme provides fast wrong password detection.

(8) The proposed scheme provides the mutual authentication. Mutual authentication means that both the user and remote system are authenticated to each other within the same protocol, while explicit key authentication is the property obtained when both implicit key authentication and key confirmation hold. As such, the proposed scheme uses the Diffie-Hellman key exchange algorithm in order to provide mutual authentication. Then, the key is explicitly authenticated by a mutual confirmation session key, $\hat{e}(P, U)^{ab}$.

(9) The proposed scheme provides perfect forward secrecy. Perfect forward secrecy means that if a long-term private key (e.g. user password PW_i or system's private key s) is compromised, this does not compromise any earlier session keys. In the proposed scheme, since the Diffie-Hellman key exchange algorithm is used to generate a session key $\hat{e}(P, U)^{ab}$, perfect forward secrecy is ensured because an attacker with a compromised system's secret key s is only able to obtain the aP and bP from an earlier session. In addition, it is also computationally infeasible to obtain the session key $\hat{e}(P, U)^{ab}$ from aP and bP, as it is a DLP and a BCDHP.

The security properties of Das et al.'s scheme and the proposed scheme are summarized in Table 1.

Table 1. A comparison of security properties

Security properties	Das et al.'s Scheme	Proposed Scheme
Passive attack	Secure	Secure
Active attack	Insecure	Secure
Guessing attack	Insecure	Secure
Stolen smart card attack	Insecure	Secure
Insider attack	Insecure	Secure
Secure password change	Not Provide	Provide
Mutual authentication	Not Provide	Provide
Session key distribution	Not Provide	Provide
Perfect forward secrecy	Not Provide	Provide
Wrong password detection	Slow	Fast
Timestamp	Required	Not Required

7 Conclusion

The current paper demonstrated that Das st al.'s scheme is vulnerable to an impersonation attack and an off-line password guessing attack. Furthermore, we presented an improved authentication scheme based on bilinear computational Diffie-Hellman problem and one-way hash function to the schemes, in order to isolate such problems. As a result, the proposed scheme is more secure than

Das et al.'s scheme and it provides mutual authentication between the user and remote system. In addition, the proposed scheme does not require time synchronization or delay-time limitations between the user and remote system. However, security of our protocol is not still proved formally. This is our future work.

Acknowledgements

This research was supported by the MIC of Korea, under the ITRC support program supervised by the IITA (IITA-2006-C1090-0603-0026).

References

1. Peyret, P., Lisimaque, G., Chua, T.Y.: Smart Cards Provide Very High Security and Flexibility in Subscribers Management. IEEE Transactions on Consumer Electronics. Vol. 36. No. 3. (1990) 744-752
2. Sternglass, D.: The Future Is in the PC Cards. IEEE Spectrum. Vol. 29. No. 6. (1992) 46-50
3. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. in: Proceedings of Algorithmic Number Theory Symposium Lecture Notes in Computer Science. Vol. 1838. Springer-Verlag. Berlin. (2000) 385-394
4. Boneh, D., Franklin, M.: Identity-based Encryption from the Weil Pairing. in: Advances in Cryptology-Crypto 2001. LNCS 2139. Springer-Verlag. (2001) 213-229
5. Smart, N.P.: An Identity Based Authentication Key Agreement Protocol Based on Pairing. Electron. Lett. Vol. 38. (2002) 630-632
6. Paterson, K.G.: ID-based Signature from Pairings on Elliptic Curves. Electron. Lett. Vol. 38. No. 18. (2002) 1025-1026
7. Wen, H.A., Lee, T.F., Hwang, T.: Provably Secure Three-party Password-based Authenticated Key Exchange Protocol Using Weil Pairing. IEE Proc.-Commun. Vol. 152. No. 2. (2005) 138-143
8. Das, M. L., Saxena, A., Gulati, V.P., Phatak, D.B.: A Novel Remote User Authentication Scheme Using Bilinear Pairings. Computers & Security. Vol. 25. No. 3. (2006) 184-189
9. Menezes, A.J., Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptograph. CRC Press. New York. (1997)
10. Ding, Y., Horster, P.: Undetectable On-line Password Guessing Attacks. ACM Operating Systems Review. Vol. 29. No. 4. (1995) 77-86
11. Menezes, A., Okamoto, T., Vanstone, S.: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory. Vol. 39. (1993) 1639-1646
12. Frey, G., Ruck, H.: A Remark Concerning m-divisibility and the Discrete Logarithm in the Divisor Class Group of Curves. Mathematics of Computation. Vol. 62. (1994) 865-874
13. Needham, R.M., Schroeder, M.D.: Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM. Vol. 21. No. 12. (1978) 993-999

Cryptanalysis of Some Proxy Signature Schemes Without Certificates[⋆]

Wun-She Yap[1], Swee-Huay Heng[2], and Bok-Min Goi[1]

[1] Centre for Cryptography and Information Security (CCIS)
Faculty of Engineering
Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
{wsyap,bmgoi}@mmu.edu.my

[2] Centre for Cryptography and Information Security (CCIS)
Faculty of Information Science and Technology
Multimedia University, Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

Abstract. The concept of proxy signature was introduced by Mambo *et al.* to delegate signing capability in the digital world. In this paper, we show that three existing proxy signature schemes without certificates, namely, the Qian and Cao identity-based proxy signature (IBPS) scheme, the Guo *et al.* IBPS scheme and the Li *et al.* certificateless proxy signature (CLPS) scheme are insecure against universal forgery. More precisely, we show that any user who has a valid public-private key pair can act as a cheating proxy signer and forge the proxy signature on behalf of the original signer at will, without obtaining the official delegation from the original signer.

Keywords: Proxy signature, identity-based, certificateless, attack.

1 Introduction

Proxy Signature and Its Applications. The concept of proxy signature was first introduced by Mambo, Usuda and Okamoto in [18]. A proxy signature scheme involved three entities, namely, the original signer, the proxy signer and the verifier [28]. A proxy signature scheme allows a designated signer called a proxy signer to sign the message on behalf of an original signer. A proxy signature convinces the verifier that the signature is signed by the proxy signer who gets the delegation right from the original signer. Proxy signatures have found various practical applications, particularly in distributed computing where delegation of rights is common. Examples include e-cash systems [19], global distribution network [2], grid computing [7] and mobile agent applications [15,16], to name a few. To illuminate how to use proxy signatures, we give more explanations on the delegating signing capabilities within organization [29]. If a manager of an

[⋆] The authors gratefully acknowledge the Malaysia IRPA grant (04-99-01-00003-EAR) and e-Science fund (01-02-01-SF0032).

organization is on leave, he has to delegate to his assistant manager the capability to sign on behalf of him.

Natural Constructions of Proxy Signature. Proxy signature can be constructed in several ways as stated in [18,14,23,25,3], according to the delegation type:

1. **Full Delegation:** The most straightforward solution is for the original signer to give its private key to the proxy signer, who can then use it to sign any messages on behalf of the original signer.
2. **Partial Delegation:** In a partial delegation scheme, a proxy signer has a new key called proxy signing key, which is different from the original signer's private key. The proxy signing key is generated by both the original signer and the proxy signer.
3. **Delegation by Certificate/Warrant:** In delegation by warrant, the original signer uses its private key and the signing algorithm of a standard signature to sign a *warrant*, which contains information regarding the particular proxy signer. After receiving the warrant, the proxy signer uses its private key and the signing algorithm of a standard signature to sign messages on behalf of the original signer.
4. **Partial Delegation with Warrant:** Kim *et al.* [14] proposed a partial delegation with warrant proxy signature scheme which enjoys the computational and bandwidth advantages over the proxy signature by warrant and the structure advantage over the proxy signature for partial delegation.

Public Key Cryptography without Certificates. Traditional public key cryptography (TPKC) was introduced by Diffie and Hellman [6] to solve the key distribution problem suffered in symmetric key cryptography. As opposed to the symmetric key cryptography, TPKC involves the use of two different keys, namely a public key and a private key, which are mathematically related to each other. However, TPKC requires the use of certificate in authenticating the public key, which leads to certificate revocation problems. Thus, the design of a secure and efficient cryptographic scheme without certificate becomes the goal of many cryptographers nowadays. Two types of public key cryptography without certificates in focus are identity-based cryptography (IBC) and certificateless public key cryptography (CLPKC).

The concept of IBC was formulated by Shamir [22] to achieve implicit certification. Shamir's original motivation was to simplify certificate management in email systems. In IBC, the public key is effectively replaced by the user's publicly available identity information or any arbitrary string which derived from the user identity (ID), thus certificate can be omitted. However, since all the private keys of the users are generated by a trusted third party (TTP) called private key generator (PKG), the private key escrow problem is inherent in the system. CLPKC [1] is a paradigm which eliminates the usage of certificates in TPKC while solving the inherent key escrow problem in IBC. CLPKC can be seen as a model that is intermediate between TPKC and IBC. In this new paradigm, the user public key is no longer any arbitrary string that identifies the user, rather, it is similar with the public key used in TPKC. The user private key is computed by using both the partial private key, a key generated by a TTP called Key Generation Centre (KGC), and the user secret value.

Our Contributions. Most of the proxy signature schemes were proposed in the public key infrastructure (PKI) setting. Recently, several proxy signature schemes adapted to IBC [28,5,21,26,24,11,10] and CLPKC [17] have also been proposed.

In this paper, we review three existing partial delegation with warrant proxy signature schemes without certificates, namely, the Qian and Cao identity-based proxy signature (IBPS) scheme [21], the Guo *et al.* IBPS [11] scheme and the Li *et al.* certificateless proxy signature (CLPS) scheme. These three schemes were derived from the provably secure identity-based signature (IBS) schemes [22,4,12]. The Qian and Cao IBPS scheme is RSA-based. RSA-based schemes are preferable since it is quite common that companies may have invested in expensive hardware and software implementations of RSA. Meanwhile, the Guo *et al.* IBPS scheme and the Li *et al.* CLPS scheme are constructed by using bilinear pairings, which is an important tool in constructing identity-based and certificateless scheme.

We show that these three schemes did not satisfy the basic security requirement of proxy signature in the ID-based setting and the certificateless setting. More precisely, we show that any user who has a valid public-private key pair can act as a cheating proxy signer and forge the proxy signature on behalf of the original signer at will, without obtaining the official delegation from the original signer.

2 Preliminaries

We review the properties of bilinear pairings below.

Bilinear Pairings: Let $(\mathbb{G}_1, \circ)$ and $(\mathbb{G}_2, \circ)$ denote two cyclic groups of prime order q ($\circ$ denotes a binary operation). A *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies the following properties:

1. Bilinearity: For all $P, Q, R \in \mathbb{G}_1$, $e(P \circ Q, R) = e(P, R)e(Q, R)$ and $e(P, Q \circ R) = e(P, Q)e(P, R)$. Thus, for any $a, b \in \mathbb{Z}_q^*$, $e(aP, bP) = e(bP, aP) = \hat{e}(P, P)^{ab}$.
2. Non-degeneracy: $e(P, Q) \neq 1_{\mathbb{G}_2}$ where $1_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$.
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

3 Cryptanalysis of the Qian and Cao IBPS Scheme

Recently, Qian and Cao proposed an IBPS scheme [21] which was derived from the Shamir IBS scheme [22]. They also proved the Shamir IBS scheme secure against adaptive chosen message attack (CMA) [8] based on the RSA assumption in the same paper. In this section, we first review the IBPS scheme proposed by Qian and Cao. We then show that this IBPS scheme is universally forgeable.

3.1 The Qian and Cao IBPS Scheme

The Qian and Cao IBPS scheme [21] is defined by the following algorithms:

1. `Setup`: The PKG runs the following steps:
   (a) Compute $n = pq$ where p and q are two large primes.
   (b) Select e at random where $\gcd(e, \phi(n)) = 1$.
   (c) Compute the `master key` d where $ed \equiv 1 \bmod \phi(n)$.
   (d) Choose $h : \{0,1\}^* \rightarrow \mathbb{Z}_{\phi(n)}$ where h is a strong one way function.
   (e) Choose $H : \{0,1\}^* \rightarrow \mathbb{Z}_n$ where H is a cryptographic hash function.
   The PKG keeps d as the `master key` and publicizes the public parameters `params` $= (n, e, h, H)$.
2. `Extract`: The user submits his ID $\in \{0,1\}^*$ to the PKG, the PKG then computes the user private key $D_{ID} = Q_{ID}^d$, where $Q_{ID} = H(ID)$. The user private key must be transmitted to the user through a secure channel. The original signer Alice has her public-private key pair as (Q_{ID_A}, D_{ID_A}), and the proxy signer Bob has his public-private key pair as (Q_{ID_B}, D_{ID_B}).
3. `Proxy Key Generation`: When Alice delegates her signing capability to the proxy signer Bob, Alice performs the following steps:
   (a) Make a warrant m_w which records the delegation policy including limits of authority, valid periods of delegation, the proxy signer ID etc.
   (b) Choose $r_A \in \mathbb{Z}_n$ at random and compute $R_A = r_A^e \bmod n$.
   (c) Compute $S_A = D_{ID_A} \cdot r_A^{h(R_A||m_w)} \bmod n$.
   (d) Send the signature $\sigma_A = (R_A, S_A)$ to the proxy signer Bob.
   After receiving the signature σ_A, Bob checks whether $S_A^e = Q_{ID_A} \cdot R_A^{h(R_A||m_w)} \bmod n$ holds. If not, Bob rejects the signature.
4. `Proxy Signature Generation`: Bob generates the proxy signature as follows:
   (a) Choose $r_B \in \mathbb{Z}_n$ at random and compute $R_B = r_B^e \bmod n$.
   (b) Compute $h = h(R_B||m_w||m)$, where m_w is the warrant and m is the message to be signed.
   (c) Compute $S_B = D_{ID_B} \cdot (r_B \cdot S_A)^{h(R_B||m_w||m)} \bmod n$.
   At last, Bob sends $\sigma_B = (R_A, R_B, S_B)$ to the verifier as a proxy signature on m_w and m for ID_A and ID_B.
5. `Proxy Signature Verification`: After receiving the σ_B, the verifier performs the following steps:
   (a) Check the warrant m_w.
   (b) Compute $Q_{ID_A} = H(ID_A)$ and $Q_{ID_B} = H(ID_B)$.
   (c) Check whether $S_B^e = Q_{ID_B} \cdot (R_B \cdot Q_{ID_A} \cdot R_A^{h(R_A||m_w)})^{h(R_B||m_w||m)} \bmod n$ holds. If not, Bob rejects the signature.

In [21], Qian and Cao showed that the Shamir IBS scheme cannot resist the blinding attack as they claimed that the user private key is the common RSA signature. The forger can pick a random $t \in \mathbb{Z}_n$ and set $\mathrm{ID}_0 = t^e \cdot$ ID mod n. The forger then requests the signature from the signer who is willing to sign on ID_0. The forger now simply computes $S = S_0 \cdot t^{-1} \bmod n$ where S and S_0 are respectively the signature for ID and ID_0 on message m. Qian and Cao thereby proposed an improved Shamir IBS scheme by setting the private key $D_{ID} = H(ID)^d$. We note that this improvement had been recommended by Shamir earlier in [22].

3.2 Attack on the Qian and Cao IBPS Scheme

Now, we show that the Qian and Cao IBPS scheme is vulnerable to the forgery attack. This strong attack is the universal forgery against no message attack where no signing oracle is required in the adversarial model. To be more precise, any user who has a valid public-private key pair can act as a cheating proxy signer (which is also considered as a forger here), to sign any message at will on behalf of the original signer, without obtaining any official delegation from the original signer. We describe the efficient algorithm that enables the forger to sign any message on behalf of the original signer. Let A denote the original signer while B denote the cheating proxy signer.

`Proxy Signature Generation`: To sign a message $m \in \{0,1\}^n$, the cheating proxy signer (the forger) who has his own private key D_{ID_B} performs the following steps:

1. Make a warrant m_w.
2. Choose $r_A \in \mathbb{Z}_n$ at random and compute $R_A = r_A^e \bmod n$.
3. Choose $r_B \in \mathbb{Z}_n$ at random and compute $R_B = r_B^e \cdot Q_{ID_A}^{-1} \bmod n$ where $Q_{ID_A} = H(ID_A)$.
4. Compute $h = h(R_B||m_w||m)$, where m_w is the warrant and m is the message to be signed.
5. Compute $S_B = D_{ID_B} \cdot (r_B \cdot r_A^{h(R_A||m_w)})^{h(R_B||m_w||m)} \bmod n$.

The forged proxy signature on message m signed by the cheating proxy signer B on behalf of the original signer A is valid since the verification step is true for the forged proxy signature as follows:

1. Check whether $S_B^e = Q_{ID_B} \cdot (R_B \cdot Q_{ID_A} \cdot R_A^{h(R_A||m_w)})^{h(R_B||m_w||m)} \bmod n$ holds. If not, rejects the signature.

$$\begin{aligned} S_B^e &= D_{ID_B}^e \cdot (r_B^e \cdot r_A^{eh(R_A||m_w)})^{h(R_B||m_w||m)} \\ &= Q_{ID_B} \cdot (r_B^e \cdot R_A^{h(R_A||m_w)})^{h(R_B||m_w||m)} \\ &= Q_{ID_B} \cdot (R_B \cdot Q_{ID_A} \cdot R_A^{h(R_A||m_w)})^{h(R_B||m_w||m)} \end{aligned}$$

   where $r_B^e = R_B \cdot Q_{ID_A}$.

The Qian and Cao IBPS scheme is therefore insecure against the universal forgery since the forger can sign any message he wants on behalf of any original signer.

4 Cryptanalysis of the Guo *et al.* IBPS Scheme

We first review the IBPS schemes proposed by Guo *et al.* [11] which was derived from the Cha and Cheon IBS scheme [4]. In [27], Yoon *et al.* showed that the Cha and Cheon IBS scheme cannot be used in constructing a provably secure identity-based aggregate signature scheme if no further modification is made. In this section, we show that the Guo *et al.* IBPS scheme is insecure against the universal forgery by using the similar approach as in Yoon *et al.*

4.1 The Guo *et al.* IBPS Scheme

The Guo *et al.* IBPS scheme [11] is defined by the following algorithms:

1. `Setup`: The PKG first chooses a security parameter k, it then chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same large prime order $q(|q| = k)$, a generator $P \in \mathbb{G}_1$ and also a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Two one way functions are also necessary: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. At last, the PKG chooses $s \in \mathbb{Z}_q^*$ as the system `master key` which is known only by itself. The PKG then computes $P_{pub} = sP$ as the system public key and publicizes the system parameters `params` $= \{\mathbb{G}_1, \mathbb{G}_2, e, q, k, P, P_{pub}, H_1, H_2\}$.
2. `Extract`: The user submits his ID $\in \{0,1\}^*$ to the PKG, the PKG then computes the user private key $D_{ID} = sQ_{ID}$, where $Q_{ID} = H_1(ID)$. The user private key must be transmitted to the user through a secure channel. The original signer Alice has her public-private key pair as (Q_{ID_A}, D_{ID_A}), and the proxy signer Bob has his public-private key pair as (Q_{ID_B}, D_{ID_B}).
3. `Proxy Key Generation`: To delegate the signing ability to the proxy signer, the original signer Alice first makes a warrant m_w, which consists of the original signer ID, the proxy signer ID, the delegation period T, the proxy signature scope, etc. Then, Alice performs some computations as follows:
   (a) Choose $x_A \in \mathbb{Z}_q^*$ at random and compute $X_{ID_A} = x_A D_{ID_A}$ and $X'_{ID_A} = x_A Q_{ID_A}$.
   (b) Compute $T = e(X'_{ID_A}, P_{pub}) = e(X_{ID_A}, P)$.
   (c) Compute $r = H_2(m_w || T || X'_{ID_A})$.
   (d) Compute $S = (x_A - r)D_{ID_A}$.
   At last, Alice sends (X'_{ID_A}, S, r) and m_w to Bob. When Bob receives the warrant m_w and (X'_{ID_A}, S, r) from Alice, he also makes some computations to check if the triple consists of the original signer's authority. Bob firstly computes:

$$\begin{aligned} T' &= e(S, P) \cdot e(rQ_{ID_A}, P_{pub}) \\ &= e(x_A D_{ID_A}, P) \\ &= e(X_{ID_A}, P) \\ &= e(X'_{ID_A}, P_{pub}) \end{aligned}$$

Then, he computes $r' = H_2(m_w||T'||X'_{ID_A})$, only if the equations $r' = r$ and $T' = e(X'_{ID_A}, P_{pub})$ are satisfied, so that Bob can confirm that he has got the original signer's authority. The proxy signature key is the combination of (D_{ID_B}, S).

4. `Proxy Signature Generation`: Bob generates the proxy signature as follows:
   (a) Choose $x_B \in \mathbb{Z}_q^*$ at random and compute $U = x_B Q_{ID_B}$.
   (b) Compute $h = H_2(m||m_w||U)$, where m_w is the warrant and m is the message to be signed.
   (c) Compute $V = S + (x_B + h)D_{ID_B}$, where S is the delegation signature from the original signer and D_{ID_B} is Bob's private key.

   At last, Bob sends $(X'_{ID_A}, U, V, m_w, m)$ to the verifier as a proxy signature.
5. `Proxy Signature Verification`: After receiving the $(X'_{ID_A}, U, V, m_w, m)$, the verifier performs the following steps:
   (a) Check the warrant m_w.
   (b) Compute $T'' = e(X'_{ID_A}, P_{pub})$.
   (c) Compute $r' = H_2(m_w||T''||X'_{ID_A})$, where m_w is the warrant.
   (d) Compute $h' = H_2(m||m_w||U)$, where m_w is the warrant and m is the message to be signed.
   (e) Check $e(P, V) = e(P_{pub}, X'_{ID_A} - r'Q_{ID_A} + U + h'Q_{ID_B})$. If it holds, $(X'_{ID_A}, U, V, m_w, m)$ will be accepted, otherwise it will be rejected.

4.2 Attack on the Guo *et al.* IBPS Scheme

Now, we show that the Guo *et al.* IBPS scheme is vulnerable to the forgery attack by using the same approach as in Yoon *et al.* Similar to our previous attack mounted on the Qian and Cao IBPS scheme, this strong attack is again the universal forgery against no message attack where no signing oracle is required in the adversarial model. More precisely, any user who has a valid public-private key pair can act as a cheating proxy signer (which is also considered as a forger here), to sign any message at will on behalf of the original signer, without obtaining any official delegation from the original signer. We describe the efficient algorithm used to sign any message on behalf of the original signer below. Let A denote the original signer while B denote the cheating proxy signer.

`Proxy Signature Generation`: To sign a message $m \in \{0,1\}^n$, the cheating proxy signer (the forger) who has his own private key D_{ID_B} performs the following steps:

1. Select a random $x_A \in \mathbb{Z}_q^*$ and compute $X'_{ID_A} = x_A Q_{ID_A}$.
2. Compute $r = H_2(m_w||T||X'_{ID_A})$ where m_w is selected at random and T is computed as $e(X'_{ID_A}, P_{pub})$.
3. Select a random $x_B \in Z_q^*$ and compute $U = x_B Q_{ID_B} - X'_{ID_A} + rQ_{ID_A}$.
4. Compute $h = H_2(m||m_w||U)$.
5. Compute $V = (x_B + h)D_{ID_B}$.
6. Return $(X'_{ID_A}, U, V, m_w, m)$ as a proxy signature.

The forged proxy signature on message m signed by the cheating proxy signer B on behalf of the original signer A is valid since the verification step is true for the forged proxy signature as follows:

1. Compute $T'' = e(X'_{ID_A}, P_{pub})$.
2. Compute $r' = H_2(m_w||T''||X'_{ID_A})$, where m_w is the warrant.
3. Compute $h' = H_2(m||m_w||U)$, where m_w is the warrant and m is the message to be signed.
4. Accept the proxy signature if $e(P, V) = e(P_{pub}, X'_{ID_A} - r'Q_{ID_A} + U + h'Q_{ID_B})$.

$$\begin{aligned} e(P,V) &= e(P_{pub}, X'_{ID_A} - r'Q_{ID_A} + U + h'Q_{ID_B}) \\ &= e(sP, X'_{ID_A} - r'Q_{ID_A} + x_BQ_{ID_B} - X'_{ID_A} + rQ_{ID_A} + h'Q_{ID_B}) \\ &= e(sP, x_BQ_{ID_B} + h'Q_{ID_B}) \\ &= e(P, (x_B + h')sQ_{ID_B}) \\ &= e(P, (x_B + h)D_{ID_B}) \end{aligned}$$

where $r' = r$ and $h' = h$.

Thus, the Guo *et al.* IBPS scheme is insecure against the universal forgery since the forger can sign any message at will on behalf of any original signer without the cooperation of the original signer at all.

5 Cryptanalysis of the Li *et al.* CLPS Scheme

The Li *et al.* CLPS scheme [17] was derived from the Cha and Cheon IBS scheme [4] and the Hess IBS scheme [12]. It is the only CLPS scheme in the literature.

5.1 The Li *et al.* CLPS Scheme

The Li *et al.* CLPS scheme [17] is defined by the following algorithms:

1. `Setup`: Given a security parameter $k \in Z^+$, the algorithm works as follows:
   (a) Generate the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
   (b) Choose an arbitrary generator $P \in \mathbb{G}_1$.
   (c) Select a random $s \in \mathbb{Z}_q^*$ and set $P_0 = sP$.
   (d) Choose a cryptographic hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.
   The system parameters are `params` $= \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, H_1, H_2 \rangle$. The message space is $M = \{0,1\}^*$. The `master key` is $s \in \mathbb{Z}_q^*$.
2. `Set-Partial-Private-Key`: Given `params` and `master-key`, this algorithm works as follows: Compute $Q_{ID_i} = H_1(ID_i) \in \mathbb{G}_1$ and output a partial private key, $D_{ID_i} = sQ_{ID_i} \in \mathbb{G}_1$. Thus, the original signer Alice has her public-private key pair as (Q_{ID_A}, D_{ID_A}), and the proxy signer Bob has his public-private key pair as (Q_{ID_B}, D_{ID_B}).

3. `Set-Secret-Value`: Given `params`, select a random value $x_{ID_i} \in \mathbb{Z}_q^*$ where x_{ID_i} is the secret value.
4. `Set-Private-Key`: Set private key, $S_{ID_i} = x_{ID_i} D_{ID_i}$.
5. `Set-Public-Key`: Given `params` and the secret value $x_{ID_i} \in \mathbb{Z}_q^*$, this algorithm computes $X_{ID_i} = x_{ID_i} P \in \mathbb{G}_1$ and $Y_{ID_i} = x_{ID_i} P_0 \in \mathbb{G}_1$.
6. `Generation of the Proxy Key`: To delegate the signing ability to the proxy signer, the original signer Alice makes a warrant m_w first, which consists of the original signer ID, the proxy signer ID, the delegation period T, the proxy signature scope, etc. Then, Alice makes some computations as follows:
   (a) Choose $r \in \mathbb{Z}_q^*$ at random and compute $U = rQ_{ID_A}$.
   (b) Compute $h_A = H_2(m_w||U)$.
   (c) Compute $V = (r + h_A)S_{ID_A}$.
   At last, Alice sends (U, V) and m_w to Bob. When Bob receives the warrant m_w and (U, V) from Alice, he performs the following steps:
   (a) Check whether $e(X_{ID_A}, P_0) = e(Y_{ID_A}, P)$ holds.
   (b) Compute $h_A = H_2(m_w||U)$.
   (c) Check whether $e(P, V) = e(Y_{ID_A}, U + h_A Q_{ID_A})$ holds.
   Then, the proxy signature key S_P is computed as $S_P = V + S_{ID_B}$.
7. `Proxy Signature Generation`: Bob can generate the proxy signature as follows:
   (a) Choose $a \in \mathbb{Z}_q^*$ at random and compute $R = e(P, P)^a$.
   (b) Compute $h_B = H_2(m||R)$, where m is the message to be signed.
   (c) Compute $S = h_B S_P + aP$.
   At last, Bob sends the proxy signature (R, U, S, m_w, m) to the verifier.
8. `Proxy Signature Verification`: After receiving (R, U, S, m_w, m), the verifier performs the following steps:
   (a) Check whether $e(X_{ID_A}, P_0) = e(Y_{ID_A}, P)$ holds.
   (b) Check whether $e(X_{ID_B}, P_0) = e(Y_{ID_B}, P)$ holds.
   (c) Compute $R' = e(P, S)e(Y_{ID_A}, -h_B(U + h_A Q_{ID_A}))e(Y_{ID_B}, -h_B Q_{ID_B})$, where $h_A = H_2(m_w||U)$ and $h_B = H_2(m||R)$.
   (d) Accept the proxy signature if and only if $h_B = H_2(m||R')$.

5.2 Attack on the Li *et al.* CLPS Scheme

Now, we show that the Li *et al.* certificateless proxy signature scheme is in fact vulnerable to the public key replacement attack against the Type I adversary. Recall that Type I adversary does not possess the knowledge of the `master key` s, but the adversary can perform public key replacement, i.e. replacing the public key with its choice. This attack is essentially the similar attack mounted by Huang *et al.* [13] against the Al-Riyami and Paterson CLS scheme [1]. More precisely, this strong attack is the universal forgery against no message attack where no signing oracle is required in the Type I adversarial model and the forger can sign any message at will.

We now describe the efficient algorithm used to mount the public key replacement attack against the Li *et al.* CLPS scheme below. This efficient algorithm enables the forger to sign any message at will.

`Sign`: To sign a message m and a warrant m_w on identities ID_A and ID_B, the Type I adversary performs the following steps:

1. Select a random $U, S \in \mathbb{G}_1$ and compute $h_A = H_2(m_w||U)$.
2. Select a random $r \in \mathbb{Z}_q^*$.
3. Compute $R = e(P,S)e(P_0, -(U + h_A Q_{ID_A}))e(rP_0, -Q_{ID_B})$.
4. Compute $h_B = H_2(m, R)$.
5. Set $x_{ID_A} = h_B^{-1} \in \mathbb{Z}_q^*$ and $x_{ID_B} = h_B^{-1} \cdot r \in \mathbb{Z}_q^*$.
6. Compute $X'_{ID_A} = x_{ID_A}P$, $Y'_{ID_A} = x_{ID_A}P_0$, $X'_{ID_B} = x_{ID_B}P$, $Y'_{ID_B} = x_{ID_B}P_0$.
7. Replace the user public key with $\langle X'_{ID_A}, Y'_{ID_A}, X'_{ID_B}, Y'_{ID_B} \rangle$.
8. Return the proxy signature (R, U, S, m_w, m).

The forged signature of message m and warrant m_w on identities ID_A and ID_B is valid the forged signature can be verified as follows:

1. Check whether $e(X'_{ID_A}, P_0) = e(Y'_{ID_A}, P)$ and $e(X'_{ID_B}, P_0) = e(Y'_{ID_B}, P)$ hold. If not, return *Error* and abort the verification. Notice that

$$\begin{aligned} e(X'_{ID_A}, P_0) &= e(x_{ID_A}P, sP) \\ &= e(x_{ID_A}sP, P) \\ &= e(Y'_{ID_A}, P) \end{aligned}$$

$$\begin{aligned} e(X'_{ID_B}, P_0) &= e(x_{ID_B}P, sP) \\ &= e(x_{ID_B}sP, P) \\ &= e(Y'_{ID_B}, P) \end{aligned}$$

2. Compute $R' = e(P,S)e(Y_{ID_A}, -h_B(U + h_A Q_{ID_A}))e(Y_{ID_B}, -h_B Q_{ID_B})$.
3. Accept the signature if and only if $h_B = H_2(m, R')$ holds.

$$\begin{aligned} R' &= e(P,S)e(Y_{ID_A}, -h_B(U + h_A Q_{ID_A}))e(Y_{ID_B}, -h_B Q_{ID_B}) \\ &= e(P,S)e(x_{ID_A}P_0, -h_B(U + h_A Q_{ID_A}))e(x_{ID_B}P_0, -h_B Q_{ID_B}) \\ &= e(P,S)e(h_B^{-1}P_0, -(U + h_A Q_{ID_A}))^{h_B} e(h_B^{-1}rP_0, -Q_{ID_B})^{h_B} \\ &= e(P,S)e(P_0, -(U + h_A Q_{ID_A}))^{h_B \cdot h_B^{-1}} e(rP_0, -Q_{ID_B})^{h_B \cdot h_B^{-1}} \\ &= e(P,S)e(P_0, -(U + h_A Q_{ID_A}))e(rP_0, -Q_{ID_B}) \\ &= R \end{aligned}$$

Since $R' = R$ holds, then $h_B = H_2(M, R')$ holds too.

The public key of ID_B is different from the public key of ID_A since a random r is included.

6 Conclusion

We mounted some attacks on three proxy signature schemes without certificates, they are the Qian and Cao IBPS scheme, the Guo *et al.* IBPS scheme and the

Li *et al.* CLPS scheme. From the above security analyses, we may conclude that the security of a proxy signature scheme deriving from a signature scheme is not guaranteed even though the underlying signature scheme is provably secure. Thus, extra caution must be exercised in extracting this kind of scheme.

References

1. S.S. Al-Riyami and K.G. Paterson. Certificateless Public Key Cryptography. *In Proceedings of ASIACRYPT 2003*, LNCS 2894, pp. 452-473, Springer-Verlag, 2003.
2. A. Bakker, M. Steen and A.S. Tanenbaum. A Law-abiding Peer-To-Peer Network for Free-Software Distribution. *In Proceedings of NCA 2001*, pp. 60-67, IEEE, 2001.
3. A. Boldyreva, A. Palacio and B. Warinschi. Secure Proxy Signature Schemes for Delegation of Signing Rights. *Cryptography ePrint Archive*, http://eprint.iacr.org/2003/096.
4. J. Cha and J. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. *In Proceedings of PKC 2003*, LNCS 2567, pp. 18-30, Springer-Verlag, 2003.
5. S.S.M. Chow, R.W.C. Liu, L.C.K. Hui and S.M. Yiu. Identity-Based Delegation Network. *In Proceedings of Mycrypt 2005*, LNCS 3715, pp. 99-115, Springer-Verlag, 2005.
6. W. Diffie and M. Hellman. New Directions in Cryptography.. *IEEE Transactions on Information Theory*, 22(6), pp. 644-654, 1976.
7. I. Foster, C. Kesselman, G. Tsudik and S. Tuecke. A Security Atchitecture for Computational Grids. *In Proceedings of CCS 1998*, pp. 83-92, ACM Press, 1998.
8. S. Goldwasser, S. Micali and R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 281–308, 1988.
9. C. Gu and Y. Zhu. Provable Security of ID-Based Proxy Signature Schemes. *In Proceedings of ICCNMC 2005*, LNCS 3619, pp. 1277-1286, Springer-Verlag, 2005.
10. C. Gu and Y. Zhu. An Efficient ID-Based Proxy Signature Scheme from Pairings. *Cryptology ePrint Archive*, http://eprint.iacr.org/2006/158.
11. S. Guo, Z. Cao and R. Lu. An Efficient ID-Based Multi-Proxy Multi-Signature Scheme. *In Proceedings of IMSCCS 2006*, Volume 2, pp. 81-88, IEEE, 2006.
12. F. Hess. Efficient Identity Based Signature Schemes based on Pairings. *In Proceedings of SAC 2003*, LNCS 2595, pp. 310-324, Springer-Verlag, 2003.
13. X. Huang, W. Susilo, Y. Mu and F. Zhang. On the Security of Certificateless Signature Schemes from Asiacrypt 2003. *In Proceedings of CANS 2005*, LNCS 3810, pp. 13-25, Springer-Verlag, 2005.
14. S. Kim, S. Park and D. Won. Proxy Signatures, Revisited. *In Proceedings of ICICS 1997*, LNCS 1334, pp. 223-232, Springer-Verlag, 1997.
15. B. Lee, H. Kim and K. Kim. Strong Proxy Signature and Its Applications. *In Proceedings of SCIS 2001*, Vol. 2/2, pp. 603-608, 2001.
16. B. Lee, H. Kim and K. Kim. Secure Mobile Agent Using Strong Non-Designated Proxy Signature. *In Proceedings of ACISP 2001*, LNCS 2119, pp. 474-486, Springer-Verlag, 2001.
17. X. Li, K. Chen and L. Sun. Certificateless Signature and Proxy Signature Schemes from Bilinear Pairings. *Lithuanian Mathematical Journal*, Vol 45(1), pp. 76-83, Springer-Verlag, 2005.

18. M. Mambo, K. Usuda and E. Okamoto. Proxy Signatures: Delegation of the Power to Sign Messages. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E79-A, No 9, pp. 1338-1354, 1996.
19. T. Okamoto, M. Tada and E. Okamoto. Extended Proxy Signatures for Smart Cards. *In Proceedings of ISW 1995*, LNCS 1729, pp. 247-258, Springer-Verlag, 1999.
20. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. *In Proceedings of EUROCRYPT 1996*, LNCS 1070, pp. 387-398, Springer-Verlag, 1996.
21. H. Qian and Z. Cao. A Novel ID-Based Partial Delegation with Warrant Proxy Signature Scheme. *In Proceedings of ISPA 2005*, LNCS 3759, pp. 323-331, Springer-Verlag, 2005.
22. A. Shamir. Identity Based Cryptosystems and Signature Scheme. *In Proceedings of CRYPTO 1984*, LNCS 196, pp. 47-53, Springer-Verlag, 1984.
23. H.−M. Sun and B.−T. Hsieh. On the Security of Some Proxy Signature Schemes. *Cryptology ePrint Archive*, http://eprint.iacr.org/2003/068.
24. Q. Wang and Z. Cao. Efficient ID-Based Proxy Signature and Proxy Signcryption from Bilinear Pairings. *In Proceedings of CIS 2005*, LNAI 3802, pp. 167-172, Springer-Verlag, 2005.
25. G. Wang, F. Bao, J. Zhou and R.H. Deng. Security Analysis of Some Proxy Signatures. *In Proceedings of ICISC 2003*, LNCS 2971, pp. 305-319, Springer-Verlag, 1999.
26. J. Xu, Z. Zhang and D. Feng. ID-Based Proxy Signature Using Bilinear Pairings. *In Proceedings of ISPA 2005*, LNCS 3759, pp. 359-367, Springer-Verlag, 2005.
27. H.J. Yoon, J.H. Cheon and Y. Kim. A New Identity-Based Signature Scheme with Batch Verification. *In Proceedings of ICISC 2004*, LNCS 3506, pp. 233-248, Springer-Verlag, 2004.
28. F. Zhang and K. Kim. Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairing. *In Proceedings of ACISP 2003*, LNCS 2727, pp. 312-323, Springer-Verlag, 2003.
29. K. Zhang. Threshold Proxy Signature Schemes. *In Proceedings of ISW 1997*, LNCS 1396, pp. 272-290, Springer-Verlag, 1997.

Performance Evaluation of Java Card Bytecodes

Pierre Paradinas[1], Julien Cordry[2], and Samia Bouzefrane[2]

[1] INRIA Rocquencourt 78150 Le Chesnay France
Pierre.Paradinas@inria.fr
[2] CNAM 292 rue Saint-Martin 75003 Paris France
firstname.lastname@cnam.fr

Abstract. The advent of the Java Card standard has been a major turning point in smart card technology. With the growing acceptance of this standard, understanding the performance behaviour of these platforms is becoming crucial. To meet this need, we present in this paper, a benchmark framework that enables performance evaluation at the bytecode level. The first experimental results show that bytecode execution time isolation is possible.

Keywords: Java Card, Benchmark, Performance.

1 Introduction

With more than one billion copies per year, smart cards are an important device of today's information society. The development of the Java Card standard made this device even more popular: capable of processing a subset of the platform independent, object oriented, and widely used programming language Java, the Java Card puts smart card technology at the disposal of many programmers and significantly shortens the time to market for smart card applications [3]. Moreover, cards are "open platforms" in the sense that programs (applets) can be added, that is, uploaded and executed on the platforms.

In this context, understanding the performance behaviour of Java Card platforms is important to the Java Card community (users, smart card manufacturers, card software providers, card users, card integrators, etc.). Currently, there is no solution on the market which makes it possible to evaluate the performance of a smart card that implements Java Card technology. In fact, the programs which realize this type of evaluations are generally proprietary and not available to the whole of the Java Card community. Hence, the only existing and published benchmarks are used within research laboratories (e.g., SCCB project from CEDRIC laboratory [7,8] or IBM Research [13]). However, benchmarks are important in the smart card area. Indeed, from smart card manufacturers point of view, standards will be more and more important in the smart card industry, as it is the case currently for the information technology domain. Furthermore, it is of primary importance to differentiate the products of the companies especially when the products are standardized. From a smart card customers point of view, benchmarks allow to understand the platform performance in terms of evaluation and prediction. It helps choose a service according to its QoS, its execution time or its memory consumption. Besides, being able to efficiently measure the performance of a cryptographic device such as a smart card in terms of time, memory or power consumption might be used to perform some security attacks and evaluations.

In this paper, we propose a general benchmarking solution to establish the execution time of Java Card bytecodes. We show that our proposed solution allows us to ascertain the feasibility of bytecode execution time isolation. Here, we restrict ourselves to presenting the comparative performances of arithmetic operations on several smart cards.

The remainder of this paper is organised as follows. In section 2, we give a brief introduction of the Java Card technology and present some related benchmarking solutions. Section 3 presents our benchmark framework and describes a general solution to achieve bytecode execution time isolation. We explain in section 4 how we revise the general solution to suit arithmetic operations benchmarking. Section 5 shows the results pertaining to arithmetic performance and also presents how the stability of the result is a function of the execution frequency of the bytecode under analysis. We present some future works in section 6 and conclude in section 7.

2 Java Card and Benchmarking

2.1 Java Card Technology

Java Card technology provides means of programming smart cards [6,1] with a subset of the Java programming language. Today's smart cards are small computers, providing 8, 16 or 32 bits CPU with clock speeds ranging from 5 up to 40MHz, ROM memory between 32 and 64KB, EEPROM memory (writable, persistent) between 16 and 32KB and RAM memory (writable, non-persistent) between 1 and 4KB. Smart cards communicate with the rest of the world through application protocol data units (APDUs, ISO 7816-4 standard). The communication is done in master-slave mode. It is always the terminal application that initializes the communication by sending the command APDU to the card and then the card replies by sending a response APDU (possibly with empty contents). In case of Java powered smart cards, besides, the operating system, the card's ROM contains a Java Card Virtual Machine (JCVM) which implements a subset of the Java programming language and allows Java Card applets to run on the card.

A Java Card applet should implement the `install` method responsible for the initialization of the applet (usually it just calls the applet constructor) and a `process` method for handling incoming command APDUs and sending the response APDUs back to the host. There can be more than one applet existing on a single card, but there can be only one active at a time (the active one is the most recently selected by the Java Card Runtime Environment – JCRE). A normal Java compiler is used to convert the source code into Java bytecodes. Then a converter must be used to convert the bytecode into a more condensed form (CAP format) that can be loaded onto a smart card. The converter also checks that no unsupported features (like floats, strings, etc.) are used in the bytecode. This is sometimes called off-card or off-line bytecode verification.

2.2 Some Attempts for Measuring Java Card Performance

Currently, there is no standard benchmark suite which can be used to demonstrate the use of the JCVM and to provide metrics for comparing Java Card platforms, thus allowing the Java Card users to take decision about which environments are most suitable for their needs. In fact, even if numerous benchmarks have been developed around the JVM, there are few works that attempt to evaluate the performance of smart cards.

For example, SCCB (Smart Card CNAM Benchmark) from CEDRIC laboratory [7,8]), was initially a very ambitious project which aimed at measuring the performance of Java Card platforms. Unfortunately, the results obtained during experiments were not accurate because the measurements were initiated at the Java Card language level and were neglecting the basic operations defined at the bytecode level.

Another interesting work is that carried out by the IBM BlueZ secure systems group and concretized through a Master thesis [13]. JCOP framework has been used to perform a series of tests to cover the communication overhead, DES performance and reading and writing operations into the card's memory (RAM and EEPROM).

Markantonakis in [10] presents some performance comparisons between the two most widely used terminal APIs, namely PC/SC and OCF. He measures some operations such as: connecting/disconnecting to the smart card reader, selecting the smart card application, sending APDUs, etc.

Guyot et al. in [9] describe how to handle session mobility by storing session information in smart card. In this special context, they evaluate the performance of smart cards by implementing real services and by observing how fast the cards could retrieve and suspend a given session.

Papapanagiotou et al. in [12] evaluate the performance of two online certificate revocation and validation protocols on two different Java Card platforms in order to determine which protocol is more efficient for smart card use.

Chaumette et al. in [4,2] show the performance of a Java Card grid with respect to the scalability of the grid and with different types of cards.

Regarding the problem that we address here, the works of Guyot et al. and Papapanagiotou et al. are used in particular contexts and do not deal with Java Card platforms while Chaumette et al. deal with the performance of a grid rather than that of a single smart card.

3 General Benchmarking Framework

3.1 Introduction

Our research work falls under the MESURE project [11], a project funded by the French administration (ANR), which aims at developing a set of open source tools to measure the performance of Java Card platforms. Currently, we have developed benchmarks covering some VM related characteristics, such as arithmetic. In this paper, we have chosen to present only the results pertaining to the arithmetic benchmarks because they are completely finished, compared to others (memory specific operations for example). The benchmarks have been developed under the Eclipse environment based on JDK 1.6, with JSR268. The underlying ISO 7816 smart card architecture forces us to measure the time a Java Card platform takes to answer to a command APDU, and to use that measure to deduce the execution time of some bytecodes. The benchmarking development tool covers two parts: the script part and the applet part. The script part, entirely written in Java, defines an abstract class that is used as a template to derive test cases characterized by relevant measuring parameters such as, the operation type to measure, the number of loops, etc. A method `run()` is executed in each script to interact with the corresponding test case within the applet. Similarly, on the card is defined an abstract class that defines three methods:

- a method `setUp()` to perform any memory allocation needed during the lifetime test case.
- a method `run()` used to launch the tests corresponding to the test case of interest, and
- a method `cleanUp()` used after the test is done to perform any clean-up.

The testing applet is capable of recognizing all the test cases and to launch a particular test by executing its `run` method.

Our Eclipse environment integrates the Converter tool from Sun MicroSystems, which is used to convert a standard Java applet class into a JCA file during a first step. This file is completed pseudo-automatically by integrating the operations to be tested with the Java Card Assembly instructions, as we explain in the following paragraph. The second step consists in capgenerating the JCA file into a CAP file, so that the applet could be installed on any Java Card platform.

3.2 Isolating Bytecode Execution Time

Benchmarking bytecodes within Java Card platforms requires some subtle means in order to obtain execution results that reflect as accurately as possible the actual execution time of the isolated execution time of an arithmetic bytecode. This is because there exists a significant and non-predictable elapse of time between the beginning of the measure, characterized by the starting of the timer on the computer, and the actual execution of the bytecode of interest. This is also the case the other way round. Indeed, when performing a request on the card, the execution call has to travel several software and hardware layers down to the card's hardware and up to the card's VM (vice versa upon response). This non-predictability is mainly dependent on hardware characteristics of the benchmark environment (such as the card acceptance device (CAD), PC's hardware, etc), the OS level interferences, services and also on the PC's VM.

To minimize the effect of these interferences, we need to isolate the execution time of the bytecodes of interest, while ensuring that their execution time is sufficiently important to be measurable.

The maximization of the bytecodes execution time requires a test applet structure with a loop having a large upper bound, which will execute the bytecodes for a substantial amount of time. On the other hand, to achieve execution time isolation, we need to compute the isolated execution time of any auxiliary bytecode upon which the bytecode of interest is dependent. For example if `sadd` is the bytecode of interest, then the bytecodes that need to be executed prior to its execution are those in charge of loading its operands onto the stack, like two `sspush`. Thereafter we subtract the execution time of an empty loop and the execution time of the auxiliary bytecodes from that of the bytecode of interest to obtain the isolated execution time of the bytecode. As presented in figure 1, the actual test is performed within a method (`run`) to ensure that the stack is freed after each invocation, thus guaranteeing memory availability.

Applet framework	Test Case
`process() {`	`run() {`
`  i = 0`	`  op`$_1$
`  While i <= L`	`  op`$_2$
`  DO {`	$\vdots$
`    run()`	`  op`$_n$
`    i = i+1`	`  op`$_0$
`  }`	`}`
`}`	

Fig. 1. Test framework for a bytecode op_0

In figure 1 :

- L represents the chosen loop upper bound;
- op_0 represents the bytecode of interest;
- op_i for $i \in [1..n]$ represents the auxiliary bytecodes necessary to perform the bytecode op_0.

To compute the mean isolated execution time of op_0 we need to perform the following calculation:

$$\overline{M(op_0)} = \frac{\overline{m_L(op_0)} - \overline{m_L(Emptyloop)}}{L} - \sum_{i=1}^{n} \overline{M(op_i)}$$

Where :

- $\overline{M(op_i)}$ is the mean isolated execution time of the bytecode op_i.
- $m_L(op_i)$ is the mean global execution time of the bytecode op_i, including interferences coming from other operations performed during the measurement, both on the card and on the computer, with respect to a loop size L.

These other operations represent for example auxiliary bytecodes needed to execute the bytecode of interest, or OS and JVM specific operations. The mean is computed over a significant number of tests. It is the only value that is experimentally measured.
- *Emptyloop* represents the execution of a case where the `run` method does nothing.

The formula presented above implies that prior to computing $\overline{M(op_0)}$ we need to compute $\overline{M(op_i)}$ for $i \in [1..n]$.

4 Arithmetics

The benchmarking of arithmetic operations requires some fine-tunings. As arithmetic operations take in general a negligible amount of time to execute, the proposed general solution may not give satisfying results. More precisely, though having a large upper bound L ensures a certain degree of accuracy in our measurements, this involves making some very long and unpractical measurements. Whereas, if we perform our tests with a smaller upper bound, we can still have some cases where $\overline{m(op_i)} < \overline{m(Emptyloop)}$ (due to the small execution time of arithmetic operations), which are mainly due to sudden load changes within the benchmark platform. Consequently, this can affect adversely the mean execution time of the test. To minimize these undesirable situations, our solution, consists in executing repeatedly as many times as possible the bytecode of interest within the `run` method. Some smart cards might perform some security countermeasures (see [5]) that will degrade the execution time of multiple similar bytecodes executed in a row. In that case, our general solution will still work but we will need to perform a very large number of loops which will make the overall test tedious.

However, in the case of arithmetic operations, increasing the number of executions of the arithmetic bytecode by k times does not necessarily entail k executions of its auxiliary bytecodes ($op_1...op_n$) (generally `sspush` operations). This is due to the fact that an arithmetic operation always ends up pushing an operand onto the stack, corresponding to its result. Therefore, we can take advantage of this to optimize the overall benchmark execution time. When benchmarking an arithmetic operation requiring two operands, for instance a `sadd` operation, the `run` method will contain k executions of `sadd`, preceded by only $k+1$ executions of `sspush`, instead of $2k$. See figure 2.

The computation of the mean isolated execution time for a binary arithmetic operation op_0, when taking into account k executions of op_0 for every iteration, is presented as follows :

$$\overline{M(op_0)} = \frac{\overline{m_L(op_0)} - \overline{m_L(Emptyloop)}}{L \times k} - (k+1) \times \overline{M(\texttt{sspush num})}$$

Where :

$$\overline{M(\texttt{sspush num})} = \frac{\overline{m_L(\texttt{sspush num}))} - \overline{m_L(Emptyloop)}}{L \times k}$$

`run` method to isolate op_0	`run` method to isolate `sspush`
`run(){`	`run(){`
`sspush num`	`sspush num`
`(k)` $\left\{\begin{array}{l}\texttt{sspush num}\\ \texttt{op}_0\end{array}\right.$	`(k) {sspush num`
`}`	`}`

Fig. 2. `run` methods for binary arithmetic operations

5 Performance Results

5.1 Arithmetic Performance

We have evaluated the arithmetic performance of three Java Card 2.2 platforms denoted respectively 3060, 4045 and 2046. Cards 3060 and 2046 were designed respectively in 2006 and 2004 by the same manufacturer, whereas card 4045 was manufactured in 2004 by another provider. The benchmarks have been carried out by measuring the execution time of distinct arithmetic operations for each Java Card platform. The results presented in figure 3 show the isolated execution time of some arithmetic operations. As we can notice, the `sadd` bytecode for each card is approximately similar in time to the `ssub`, `sor`, `sand` and `sxor` bytecodes, which is normal since they are similar binary operations. We can also observe that `sneg` is the fastest one since it needs only one operand.

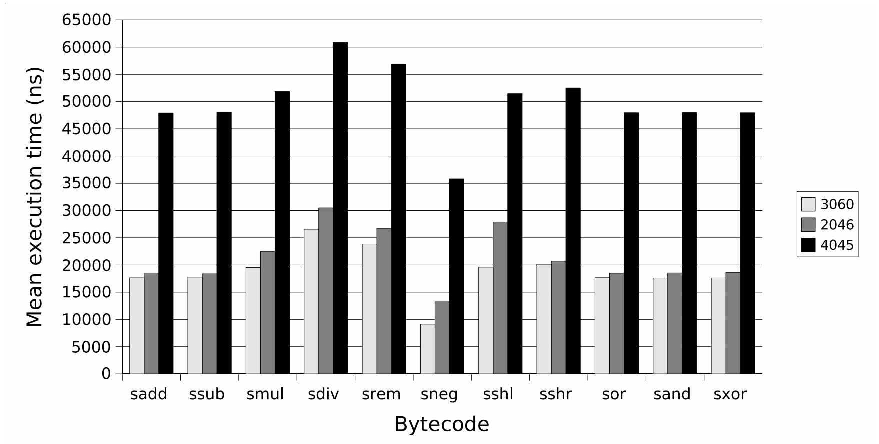


Fig. 3. Arithmetic performances of three cards

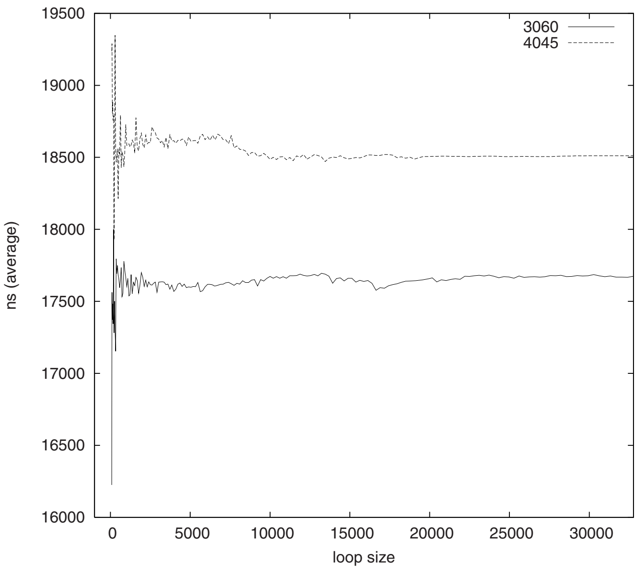


Fig. 4. Linearity evaluation on $\overline{M(\texttt{sadd})}$

With this test we have also been able to assess the characteristics of the three Java Card platforms vis-à-vis the arithmetic implementations of their JCVM. For instance, we can observe that for the cards 3060 and 4045, the `sshl` bytecode has nearly the same execution time as that of `smul` or `sshr`, whereas in card 2046 its execution time is nearer to that of `sdiv` or `srem`. From this observation, we can easily assume that `sshl` is implemented by a division in card 2046.

In conclusion, when comparing the performances of the three Java Card platforms, we can clearly see that card 3060 has slightly better performance than card 2046, whereas both of them outperform greatly card 4045.

5.2 Linearity of the Results

With our proposed bytecode execution time measurement solution, we expect that the mean execution time of the isolated bytecodes will stabilize after a certain loop size. We have checked for this linearity on the two cards 3060 and 4045. Figure 4 shows the mean execution time of a simple isolated `sadd` over 100 measures, based on each loop size. During this test, we have made use of the parameter `P2` of the APDU command to change the loop size.

As we can notice, the measures tend to reach a certain degree of stability as the loop size increases, though the results obtained for the two cards are dissimilar. We can also observe that their execution behaviour follows the same general pattern over the loop size range. This confirms the reliability of our proposed “bytecode execution time isolation” technique.

In general, the execution time stabilization is dependent upon factors such as the CAD, its driver, the computer OS, the CPU load during the test as well as

Instruction set					
Opcode	**Mnemonic**	**Reference loop**	**Opcode**	**Mnemonic**	**Reference loop**
0	nop	empty loop	117,118	<t>lookupsw itch	1 <t>load | 1 goto
1	aconst_null	empty loop	119-121	<t>return	
2-8	sconst_<n>	empty loop	122	return	
9-15	iconst_<n>	empty loop	123-126	getstatic_<t>	empty loop
16-20	<t1><t2>push	empty loop	127-130	putstatic_<t>	1 <t>load
21	<t>load	empty loop	131-134	getfield_<t>	1 aload
24-35	<t>load_<n>	empty loop	135-138	putfield_<t>	1 aload | 1 <t>load
36-39	<t>aload	1 aload | 1 sload	139	invokevirtual	
40-42	<t>store	1 <t>load	140	invokespecial	
43-54	<t>store_<n>	1 <t>load	141	invokestatic	
55-58	<t>astore	1 aload | 1 sload | 1 <t>load	142	invokeinterface	
59	pop	1 sload	143	new	empty loop
60	pop2	2 sload	144	new array	1 sload
61	dup	1 sload	145	anew array	1 sload
62	dup2	2 sload	146	arraylength	1 aload
63	dup_x	1 bspush | n sload	147	athrow	
64	sw ap_x	1 bspush | n sload	148	checkcast	1 aload
65-88	<t><arithmetic_operation>	2 <t>load	149	instanceof	1 aload
89,90	<t>inc	empty loop	150,151	<t>inc_w	empty loop
91-94	<t1>2<t2>	1 <t1>load	152-159	if<cond>_w	1 sload
95	icmp	2 iload	160-167	if_<t>cmp<cond>_w	2 <t>load
96-103	if<cond>	1 sload	168	goto_w	nop
104-111	if_<t>cmp<cond>	2 <t>load	169-172	getfield_<t>_w	1 aload
112	goto	nop	173-176	getfield_<t>_this	empty loop
113	jsr		177-180	putfield_<t>_w	1 aload | 1 <t>load
114	ret		181-184	putfield_<t>_this	1 <t>load
115,116	<t>tablesw itch	1 <t>load | 1 goto			

Fig. 5. Instruction Set

the card itself. For instance, in the case of the CAD, as the precision may vary from one CAD to another, the confidence in the results for a given loop size will vary. As a result, the loop size necessary to obtain an accurate and stable measure will depend generally upon the test environment, hence the needfulness for a loop size calibration prior to testing.

6 Future Works

6.1 Expanding the Test to Other Bytecodes

In the near future, we plan to expand the test to all bytecodes. Here also, our approach, at the outset, will be to track back any auxiliary bytecode necessary to satisfy the bytecode dependencies. The result of this dependency analysis is presented in figure 5. We categorise the terms upon which the bytecodes operate as follows:

$$\begin{aligned} t &::= a|b|i|s \\ n &::= m\_1|0|1|2|3|4|5 \\ cond &::= eq|ne|gt|ge|lt|le \end{aligned}$$

t represents the set of data types used, objects (a), bytes (b), integers (i) and shorts (s). n represents the allowed integer constants used. $cond$ represents the different execution conditions.

For most of the cases, the test will follow the general framework presented in section 3. However, we will still be confronted to some exceptional cases such as those presented in the grayed background cells of the table. Indeed, the execution time of these bytecodes cannot be isolated. One possible solution is to pair bytecodes execution. For instance, we can measure the execution time of a method invocation (such as `invokestatic`) and the `return` bytecode as a whole.

6.2 Other Issues

In this paper we have focused on benchmarking Java Card bytecodes. But in the next few months, our objective will be to evaluate the execution time at the Java Card API level.

With the benchmark results, obtained at the bytecode and API levels, we will also be able to evaluate the performance of several execution scenarios of applets. The evaluation will require the analysis of an applet structure both at the bytecode and API levels to establish the bytecodes and methods used as well as their frequencies. The potential execution time of a given applet will be defined as a function of the frequency of methods/bytecodes and their benchmark results.

7 Conclusion

With the wide use of Java in smart card technology, there is a need to evaluate the performance and characteristics of these platforms in order to ascertain whether they fit the requirements of the different application domains. For the time being, there is no open source benchmark solution for Java Card. The objective of our project [11] is to satisfy this need by providing a set of freely available tools, which, in the long term, will be used as a benchmark standard.

In this paper, we have proposed, through our general benchmark framework, a "bytecode execution time isolation" technique that helps us assess the execution time of a bytecode, with OS level and hardware interferences removed.

We have shown via experimental tests that our technique produces accurate results with a confidence varying with respect to the test environment used. Indeed, stability of the result is strongly dependent on the frequency of the execution of the bytecode under scrutiny. We discussed that to obtain an accurate and stable measure, there is a need to calibrate the benchmark framework prior to testing.

References

1. Java card 2.2.2 specification, April 2006.
2. Eve Atallah, Franck Darrigade, Serge Chaumette, Achraf Karray, and Damien Sauveron. A grid of java cards to deal with security demanding application domains. In *6th edition e-Smart conference & demos*, September 2005. Sophia Antipolis, Frensh Riviera.

3. Clemens H. Cap, Nico Maibaum, and Lars Heyden. Extending the data storage capabilities of a java-based smart card. In *Sixth IEEE Symposium on Computers and Communications (ISCC01)*. IEEE, 2001.
4. Serge Chaumette, Pascal Grange, Achraf Karray, Damien Sauveron, and Pierre Vignéras. Secure distributed computing on a java card grid. Technical Report 1331-04, LaBRI, Université Bordeaux 1, 2004.
5. Serge Chaumette and Damien Sauveron. Some security problems raised by open multiapplication smart cards. In *10th Nordic Workshop on Secure IT-systems: NordSec 2005*, October 2005.
6. Zhiqun Chen. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. Addison Wesley, 2000.
7. Jean-Michel Douin, Pierre Paradinas, and Cédric Pradel. Open benchmark for java card technology. In *e-Smart Conference*, September 2004.
8. Gilles Grimaud, Pierre Paradinas, and Eric Vétillard. Measuring the performance of the java card platform. Java One, May 2006.
9. Vincent Guyot, Nadia Boukhatem, and Guy Pujolle. Smart card performances to handle session mobility. In *ICI*. IFIP/IEEE, September 2005.
10. Constantinos Markantonakis. Is the performance of smart card cryptographic functions the real bottleneck? In *16th international conference on Information security: Trusted information: the new decade challenge*, volume 193, pages 77 – 91. Kluwer, 2001.
11. The MESURE project website. http://cedric.cnam.fr/MESURE.
12. Konstantinos Papapanagiotou, Constantinos Markantonakis, Qing Zhang, William G. Sirett, and Keith Mayes. On the performance of certificate revocation protocols based on a java card certificate client implementation. In *20th IFIP International Information Security Conference (Sec 2005) - Small Systems Security and Smart cards*, May 2005.
13. Karima Rehioui. Java card performance test framework, September 2005. Université de Nice, Sophia-Antipolis, IBM Research internship.

Reverse Engineering Java Card Applets Using Power Analysis

Dennis Vermoen[1,2], Marc Witteman[2], and Georgi N. Gaydadjiev[1]

[1] Computer Engineering, TU Delft, The Netherlands
georgi@ce.et.tudelft.nl
http://ce.et.tudelft.nl/~georgi
[2] Riscure BV, The Netherlands
{vermoen,witteman}@riscure.com
http://www.riscure.com

Abstract. Power analysis on smart cards is widely used to obtain information about implemented cryptographic algorithms. We propose similar methodology for Java Card applets reverse engineering. Because power analysis alone does not provide enough information, we refine our methodology by involving additional information sources. Issues like distinguishing between bytecodes performing similar tasks and reverse engineering of conditional branches and nested loops are also addressed. The proposed methodology is applied to a commercially available Java Card smart card and the results are reported. We conclude that our augmented power analysis can be successfully used to acquire information about the bytecodes executed on a Java Card smart card.

1 Introduction

Currently Java Card is the most commonly used platform for commercial smart cards. According to Sun Microsystems, Java Card technology grew from 750 million deployments in November 2004 to over 1.25 billion deployments in November 2005 [1,2]. Because smart cards are typically used in applications that require a high degree of security, it is needless to say that security of Java Card applications is very important.

Power analysis is a side channel analysis technique to acquire information about running processes on a device (such as smart cards) by monitoring the dynamic current usage. Power analysis on smart cards is commonly used to obtain information about running cryptographic algorithms [3,4,5,6].

In this paper, we introduce Java Card reverse engineering methodology by means of augmented power analysis. When a Java Card applet source code could be reverse engineered, possible vulnerabilities can be exploited. We performed our experiments on several commercially available Java Card smart cards. In this paper we will focus on only one specific smart card[1]. Experimental results for different smart cards can be found in [7]. Nevertheless, the majority of the proposed techniques are applicable in the general case.

[1] The specific brand and type of the smart card can not be disclosed.

The main contributions of this paper are:

- A methodology to analyse power consumption of Java Card applets;
- Techniques to determine a unique power profile template for each Java Card bytecode. In addition, we describe how templates can be recognised in an arbitrary power trace, in order to determine the execution trace;
- Additional information sources that can be used to reduce the number of errors in the generated trace;
- Techniques to convert the execution trace into structured Java Card bytecode source.

Due to the space limitation, readers are assumed to have some basic knowledge of Java Card technology (a good introduction can be found in [8,9]).

The rest of this paper is organised as follows. Section 2 discusses the methodology that we used. Section 3 presents the experimental results and the methodology refinements. Finally, we conclude in Section 4.

2 Methodology

In order to gain information and reverse engineer arbitrary Java Card applets, we selected a programmable Java Card smart card. A Java Card applet is compiled to bytecode using the Java compiler. For example, each addition operation as depicted in Figure 1 is compiled to the following bytecode sequence

`sload`, `sload`, `sadd`, `s2b`, `sstore`

The multiplication sequence looks similar (i.e. the `sadd` bytecode is replaced by the `smul` bytecode). Therefore, the power trace representing the power consumption variations of the applet execution is expected to show repetitions, making this applet interesting for power analysis.

```
public class TestApplet extends javacard.framework.Applet {
   public void process(javacard.framework.APDU apdu) {
      byte a = (byte) 0x04, d, p;
      byte buffer[] = apdu.getBuffer();
      short len = apdu.setIncomingAndReceive();
      d = buffer[(short)(javacard.framework.ISO7816.OFFSET_CDATA)];
      p = (byte)(a+d);
      p = (byte)(a*d);
      p = (byte)(a*d);
      p = (byte)(a+d);
      p = (byte)(a*d);
      p = (byte)(a+d);
      p = (byte)(a+d);
      p = (byte)(a*d);
   }
}
```

Fig. 1. Example Java Card applet

We can execute the above `process` method by sending an arbitrary command to the smart card when the applet is active. Our acquisition framework, which is described in Appendix A, is used to obtain a power trace from the execution of this test applet. The resulting power trace is depicted in Figure 2. This measurement was performed without any trigger delay, at low speed and the at maximal number of samples possible for the used equipment to gain a complete overview of the smart card power consumption.

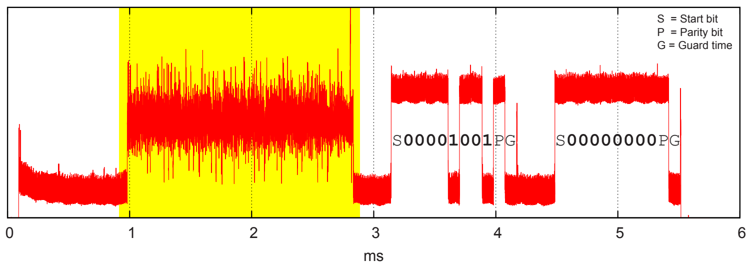


Fig. 2. Single power trace

The last part of the power trace (i.e. from 3 to 6 ms) represents the smart card response. In this case the response was `0x9000`, because the Java Card applet executed successfully. The `0x9000` response code is returned after approximately 3 ms. Therefore, the execution of the actual Java Card applet takes place in the first part of the power trace (i.e. from 1 to 3ms). Note that the power consumption increases and looks noisier during the applet execution. Possibly the investigated smart card activates some countermeasure against power analysis when the Java Card Virtual Machine (JCVM) is running. After we determined the approximate start time and duration of the Java Card applet, a larger number of power traces could be collected. By delaying the trigger signal and decreasing the number of samples, we limited the acquisition to only the interesting region of the power trace (i.e. from 1 to 3ms).

Resampling. Performing power analysis often requires the collection of a significant number of traces. When capturing 10000 traces at 200 MHz each containing 1000000 8-bit samples, the total file size becomes ≈ 9.5 GB. Resampling is a technique to reduce the total file size, at cost of losing some information. When the trace set is resampled at 4 MHz (the operating frequency of the smart card), the number of samples will be reduced by a factor 50. Each trace will then contain only 20000 samples [2] and require only 760 MB. Therefore it is advantageous to resample the traces before storing them. Some measurements that require high precision must of course not be resampled as will be shown later.

[2] All samples represent the average value of 50 samples in the original trace.

Correlation. Correlation gives a measure of association between variables [10]. It returns a value between -1 and 1, where 1 means "identical in shape" and -1 means a "inverted in shape". Correlation 0 means that the values are uncorrelated. We use correlation to recognise specific templates in a power trace. In addition, it allows us to determine if a specific input value is used by a bytecode or not. In contrast to correlation with input values, the negative correlation is not relevant when using it to recognise templates. In this paper, a correlation of 1 is represented as 100% and 0 is represented as 0%. Detailed information about the correlation function is given in Appendix B.

Averaging. As depicted in Figure 2, a single power trace is noisy. Taking the arithmetic mean of a set of traces is a simple but effective technique to remove noise. Figure 3 depicts the average of 10000 power traces of the same Java Card applet using the same input data. Note that, in contrast to Figure 2, a repeated pattern is clearly visible. The techniques described in the rest of this section assume averaged trace sets, as single traces are too noisy.

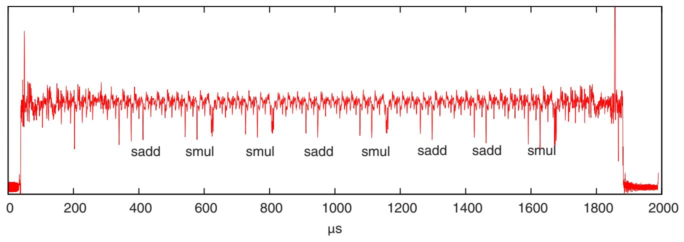


Fig. 3. Average of 10000 power traces. Note that only the interesting region (1 ms to 3 ms in Figure 2) is acquired.

Template determination. In order to recognise bytecodes in a power trace, each bytecode needs to be represented by a unique template. To determine a template for a specific bytecode, a test applet that contains this bytecode is used. For example, the source code fragment depicted in Figure 1 can be used to determine templates of 10 different bytecodes (i.e. `aload`, `baload`, `return`, `s2b`, `sadd`, `sconst_5`, `sload`, `sload_2`, `smul` and `sstore`). Storing frequently occurring bytecode sequences as a single template is also considered. These templates are referred to as *combined templates*.

The execution of the fetch, decode and execute sequence of the JCVM also corresponds to a specific template (referred to as *JCVM template*). This is advantageous, because this template can be utilised to split the power trace into separate parts representing the individual bytecodes. By comparing them with the bytecode of the known Java Card applet, it is possible to store them as the template for that specific bytecode. The same technique can also be used to determine templates of native methods, e.g. a DES operation [7]. It is important to note that the templates considered here are valid only for the specific smart card type used.

Template Recognition. The templates determined in the previous section can be used to process an unknown applet. We developed a program that automatically matches n templates against an average power trace by using the correlation technique described earlier. The result of this program is a set of n traces containing the correlation of the power trace with each template. One example is depicted in Figure 4 where the power trace (shown in red on the first row) and its correlation with templates for `sload`, `baload`, `sadd+s2b+sstore` and `smul+s2b+sstore` respectively are shown. From Figure 4 can be concluded that the bytecode sequence `smul+s2b+sstore` is probably executed three times during the applet execution.

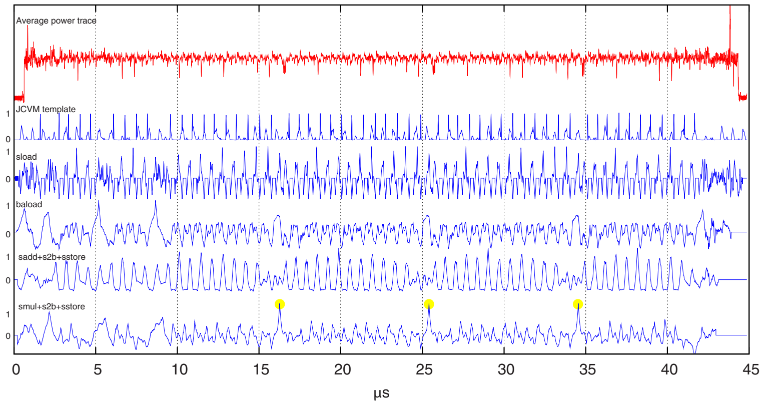


Fig. 4. Result of the template matching process

3 Experimental Results

As our first experiment, we developed a Java Card applet that performs only two addition statements. Table 1 shows the results of the template recognition process as described in Section 2. The first column contains the actual bytecodes that were executed. The second column contains the bytecode with the best correlation, while the third column contains alternative bytecode candidates that have a correlation greater than a predefined threshold (i.e. 50%). The JCVM template is used to cluster the execution trace. Note that the results contain uncertainties and even one error (i.e. on the sixth row, the `aload` bytecode has a better correlation than the actual `sload` bytecode).

Table 1. Example execution trace obtained from the power analysis

Expected	Recognised	Alternatives
sload	sload (93%)	aload (89%)
	JCVM	
sload	sload (92%)	aload (91%), sconst & sstore (57%)
	JCVM	
sadd	sadd (91%)	sload (55%), aload (51%)
	JCVM	
s2b & sstore	s2b & sstore (91%)	sload (51%)
	JCVM	
sload	sload (92%)	aload (78%), sconst & sstore (54%)
	JCVM	
sload	~~aload (92%)~~	sload (91%)
	JCVM	
sadd	sadd (90%)	sload (54%), aload (53%)
	JCVM	
s2b & sstore	s2b & sstore (90%)	sload (53%)

Our second experiment was to attempt distinguishing bytecodes that perform similar operations. Some bytecodes that are available in the JCVM are used to optimise common operations. For example, loading a short value from local variable 2 or 3 can be performed using sload_2 or sload_3 respectively. We performed this measurement at 200 MHz, because distinguishing between similar bytecodes, such as sload_2 and sload_3, is difficult using resampled traces.

We performed 12500 measurements of the power consumption during the execution of an sload_2 bytecode and another 12500 measurements during the execution of an sload_3 bytecode. Figure 5 depicts the difference between sload_2 and sload_3. There is some difference only during a small period of time (i.e. approximately 400ns). Although our experiment indicated the possibility to determine the exact type of sload operation, a lot of traces must be collected making this process very time consuming.

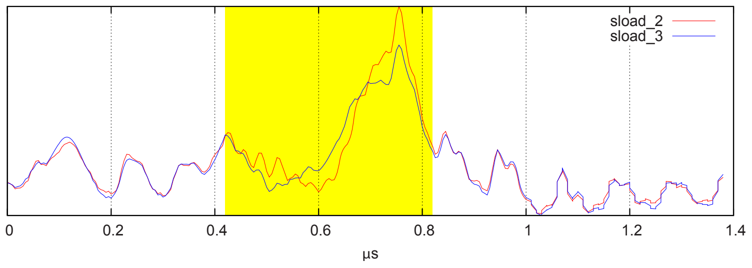


Fig. 5. Difference between sload_2 and sload_3

3.1 Methodology Refinements

Our first experiments indicated that power analysis only, sometimes can not provide enough information to recognise the correct bytecode template. We refined our methodology by identifying additional information sources as will be described in this section.

Impossible Bytecode Sequences. Not all bytecode can follow each other. During the reverse engineering process it is advantageous to keep an operand type stack. Although storing the operands themselves is difficult, storing their types is much easier. Based on the elements on top of the operand type stack, some bytecodes can be excluded from the set of possible follow-up bytecodes. Note that this approach will greatly reduce the search space.

When this technique is applied to the example of Table 1, the impossible bytecode sequence in this example: `sload`, `aload`, `sadd` can be recognised. Because an `sadd` bytecode expects a `short` on top of the operand stack, while an `aload` bytecode pushes an *objectref*, the `aload` must be replaced by an alternative bytecode (i.e. the `sload` bytecode that matches for 91%). This results in: `sload`, `sload` (first alternative), `sadd`. In this case, it is assumed that the `sload` bytecode on line 5 and the `sadd` bytecode on line 7 are correctly recognised.

Unlikely Bytecode Sequences. Besides impossible bytecode sequences, as described above, there are also bytecode sequences that are unlikely to occur even they are allowed by the JCVM. For example, `sconst_0` , `sdiv` (divide by constant 0) is obviously not likely to occur in a normal trace.

Bytecode Statistics. Statistical information about already processed Java Card applets can also be used. Because the bytecode of Java Card applets on a smart card is usually generated by the Java compiler, certain patterns will occur more often than others. For example, experiments reveal that a `for` loop, will always be generated as depicted in Figure 6. Other examples are `i=0` and `i++` which are depicted on lines 1-2 and 5-9 respectively. Saving a template for each of these frequently occurring patterns is advantageous. Experiments showed that templates which contain more samples usually have a less noisy correlation, as depicted in the last trace of Figure 4.

Input Data. Besides correlating a power trace with templates, correlation with input data contained in the command can also be used to determine which bytecode uses input data. The example in Figure 7 depicts the average power trace of the `smul` bytecode. In addition, it also depicts the correlation with the first operand of the `smul` bytecode and the correlation with a random byte, which is not used by the `smul` bytecode. From Figure 7 one can conclude that it is possible to determine if a specific input value is used by a bytecode.

```
     sconst_0
     sstore_2
     goto L2
L1:  // Loop body is inserted here
     sload_2
     sconst_1
     sadd
     s2b
     sstore_2
L2:  sload_2
     bspush    3
     if_scmplt L1
```

Fig. 6. A `for` loop as generated by the Java compiler

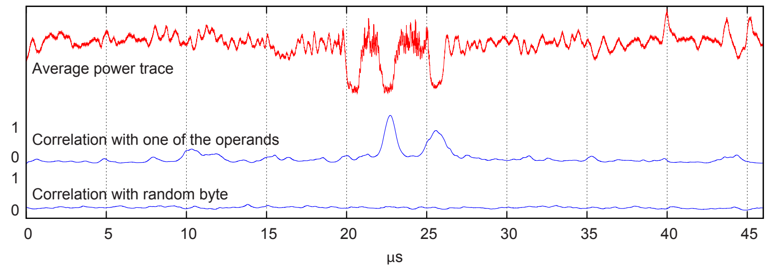


Fig. 7. Correlation between input data and the power trace

Bytecode duration. In some situations the duration of a bytecode execution gives useful information. We found that the duration of a conditional branch bytecode indicates if a branch is taken or not. For example, the duration of the non-taken `if_scmplt` bytecode is approximately 5.75µs. In case the branch is taken the duration increases by 4.5µs to 10.25µs.

Loop Rerolling. Using the techniques described earlier, it is possible to obtain an applet execution trace. In order to reverse engineer a Java Card applet completely, the execution trace should be transformed into structured bytecode. This step is certainly not trivial, because an execution trace is very likely to contain loops.

In the ideal case, the reverse engineering process would generate an execution trace as depicted in Figure 8. This figure shows the execution of a loop which is iterated 3 times.The execution trace can be divided into several parts. First of all, lines 3-6 indicate the presence of a loop. The `goto` statement is used to branch to the conditional part of the loop which loads a `short` value (`sload`), pushes a constant (`bspush`) and branches if the `short` comparison succeeds (`if_scmplt`). Second, the lines following the `goto` statement (i.e. lines 4-6) can be used to split the execution trace of the loop into repetitive parts. The end of the loop is reached when the conditional branch bytecode is not followed by the loop body.

```
1  sconst_0
2  sstore_2
3  goto
4  sload_2
5  bspush    3
6  if_scmplt
7  // Loop body
8  sload_2
9  sconst_1
10 sadd
11 s2b
12 sstore_2
13 sload_2
14 bspush    3
15 if_scmplt
16 // Loop body
17 sload_2
18 sconst_1
19 sadd
20 s2b
21 sstore_2
22 sload_2
23 bspush    3
24 if_scmplt
25 // Loop body
26 sload_2
27 sconst_1
28 sadd
29 s2b
30 sstore_2
31 sload_2
32 bspush    3
33 if_scmplt
```

Fig. 8. Execution trace of the program depicted in Figure 6

In addition, the duration of the conditional branch bytecode may also indicate the end of the loop, as explained earlier.

Besides reconstructing the loop, rerolling the loop has other advantages. First of all it is possible to derive the labels originally used on lines 3, 6, 15, 24 and 33. Second, it is very common that the same loop variable is used in the initialisation, condition and increment part of the loop. Therefore it is likely that the bytecodes on lines 2, 4, 8, 12, 13, 17, 21, 22, 26, 30 and 31 share the same local variable index.

Although this technique works fine for this relatively simple example, it is rather difficult to automate this process. Detecting a nested loop as such is not very difficult, because the nested loop will cause an additional `goto` statement. However, reconstruction of a nested loop is difficult because the conditional part may contain similar statements and the execution traces may contain errors. Moreover the loop may contain conditional statements.

Conditional Branches. Conditional branch bytecodes, such as `if_scmplt`, make the reverse engineering process more difficult. By varying the input data, it is possible that another part of the source code is executed. Without knowledge of the source code it can be difficult to determine on what input data a conditional branch bytecode is dependent. There are two ways to determine such dependency:

– Use correlation between random input data and the power profile of the conditional branch bytecode;
– Inspect the reverse engineered applet first and try to derive what input data is used in the condition.

It is however possible that a varying input data does not affect the conditional branch, for example when it is based on an internal state or data from a random generator. In this case the condition has to be determined from the partially reverse engineered source code.

3.2 Execution Trace Decompilation

When the structured bytecode is available, it is relatively easy to reconstruct source-level expressions. In [11], a technique to automatically decompile Java bytecodes into Java source code is presented. Although the referred paper focuses on decompiling standard Java bytecode, we successfully implemented a Java Card version. Implementation details of this process are outside the scope of this paper.

4 Conclusion and Future Work

In this paper we showed that power analysis can be used to acquire information about executed bytecodes on a Java Card smart card. Using the right equipment and a methodology to determine and recognise bytecode templates, we were able to generate an execution trace of a Java Card applet. Although the tested smart card activates a countermeasure against power analysis when the JCVM is active, we found that this countermeasure is not very effective. Next, we showed that besides power analysis, additional information sources can be used to reduce the number of errors and uncertainties in the execution trace based on the fact that:

– some bytecode sequences cannot occur in a valid Java Card applet;
– some bytecode sequences are very unlikely to occur, although they are valid;
– statistics of other Java Card applets can identify frequently occurring bytecode sequences.
– correlation with input data can be used to determine which variables depend on input data;
– the duration of some bytecodes can provide information. The duration of a conditional branch bytecode indicates if a branch is taken or not.

In addition, we presented techniques to generate structured bytecode from the execution trace using loop rerolling. Most of the time however, this step will be difficult, as the execution trace may also contain nested loops and other conditional statements. On the other hand it may still be possible though, to reverse engineer those parts manually.

Finally we showed that structured bytecode, once it is available, can be decompiled relatively easy to Java source code using algorithms which are also used to decompile regular Java applications.

4.1 Directions for Future Work

There are a couple of topics that will be addressed in the future. First of all, it would be interesting to see if the techniques described in this paper can also be applied to RFIDs (contactless smart cards). Second, we intend to investigate different countermeasures aimed to prevent Java Card applets from being reverse engineered using power analysis. Finally, a program that performs the template determination for all bytecodes automatically would be interesting.

References

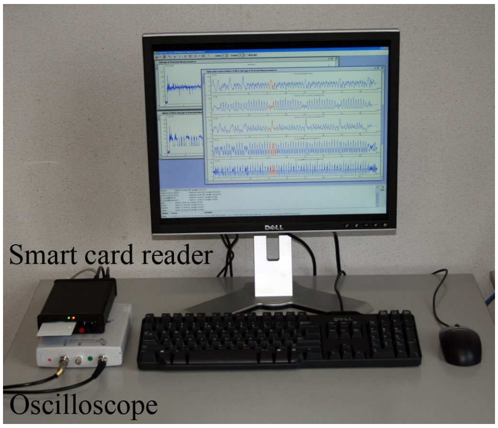
1. Sun Microsystems, Inc. http://www.sun.com/smi/Press/sunflash/2004-11/sunflash.20041102.1.xml (2004)
2. Sun Microsystems, Inc. http://www.sun.com/smi/Press/sunflash/2005-11/sunflash.20051115.2.xml (2005)
3. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In Wiener, M.J., ed.: CRYPTO. Volume 1666 of Lecture Notes in Computer Science., Springer (1999) 388–397
4. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Koblitz, N., ed.: CRYPTO. Volume 1109 of Lecture Notes in Computer Science., Springer (1996) 104–113
5. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Computers **51**(5) (2002) 541–552
6. Witteman, M.: Advances in smartcard security. Information Security Bulletin **7** (2002) 11–22 Also available at http://www.riscure.com/articles/ISB0707MW.pdf.
7. Vermoen, D.: Reverse engineering of java card applets using power analysis (2006) Available at http://ce.et.tudelft.nl/publicationfiles/1162_634_thesis_Dennis.pdf.
8. Chen, Z.: Java Card Technology for Smart Cards: Architecture and Programmer's Guide. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000)
9. Witteman, M.: Java card security. Information Security Bulletin **8** (2003) 291–298 Also available at http://www.riscure.com/articles/ISB0808MW.pdf.
10. Press, W., Teukolsky, S., Vettering, W., Flannery, B.: Numerical Recipes in C++ – Second Edition. Cambridge University Press, Cambridge, UK (2002)
11. Proebsting, T.A., Watterson, S.A.: Krakatoa: Decompilation in java (does bytecode reveal source?). In: COOTS, USENIX (1997) 185–198

A Acquisition Framework

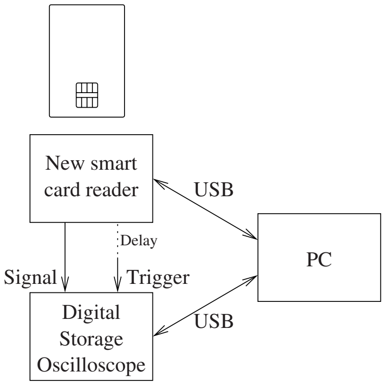
In order to collect power traces, we developed an acquisition system. As depicted in Figure 9, the system consists of a smart card reader, a Digital Storage Oscilloscope (DSO) and a PC. Both the smart card reader and the DSO are connected to the PC using separate USB channels.

Our initial system triggered the oscilloscope using software. Unfortunately this caused the oscilloscope to be triggered at different positions in the applet execution. The time required to execute a Java Card applet, is typically longer than an oscilloscope can store in its memory. Therefore we developed a new smart card reader that automatically triggers the oscilloscope after sending the last byte of a command APDU. The trigger signal can eventually be delayed with µs precision to inspect different parts of applet under test.

The experiments performed in this paper are performed using a 200 MHz DSO that can store approximately one million samples.



(a)



(b)

Fig. 9. The acquisition system

B Correlation Formulas

In order to understand how the correlation between two variables can be computed, some other functions must be defined first. The variance of x is defined as:

$$var(x) = \frac{(\sum x_i - \overline{x})^2}{n-1} \tag{1}$$

where x_i represents the i-th element of x, $\overline{x}$ is the algebraic mean of x, and n is the size of x. The covariance of x and y provides a measure of how much x and y are related and is defined as:

$$cov(x, y) = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{n-1} \tag{2}$$

The covariance is difficult to interpret though, because it depends on the scale of the input values. A better measure, independent on the absolute values of the input is given by the correlation function which is defined as:

$$corr(x, y) = \frac{cov(x, y)}{\sqrt{var(x) \cdot var(y)}} \tag{3}$$

An Embedded System for Practical Security Analysis of Contactless Smartcards

Timo Kasper, Dario Carluccio, and Christof Paar

Communication Security Group,
Ruhr-University Bochum, Germany
{tkasper,carluccio,cpaar}@crypto.rub.de
www.crypto.rub.de

Abstract. ISO 14443 compliant smartcards are widely-used in privacy and security sensitive applications. Due to the contactless interface, they can be activated and read out from a distance. Thus, relay and other attacks are feasible, even without the owner noticing it. Tools being able to perform these attacks and carry out security analyses need to be developed. In this contribution, an implementation of a cost-effective, freely programmable ISO 14443 compliant multi function RFID reader and fake transponder is presented that can be employed for several promising purposes.

Keywords: RFID, Low Level Reader, Fake RFID Tag, Relay Attack.

1 Introduction

As technology evolves and chip sizes decrease, RFID (Radio Frequency Identification) is becoming widely-used for ubiquitous tasks. The ISO 14443 [14] norm for contactless smartcards is currently employed in various security sensitive applications, such as the electronic passport [3] to store biometric data and RFID-enabled credit cards [31]. The contactless interface brings new opportunities for potential attackers: The device can not only be activated and read out without the actual owner taking note of it, but also can the transmission of data via the RF (Radio Frequency) field be eavesdropped from a distance of several meters [8]. This demanded for countermeasures, such as encryption of the interchanged data and the BAC (Basic Access Control) in the electronic passport [15].

New Perils. However, modern attackers get physical access to the chip or its electromagnetic field and perform so called side channel attacks like a **DEMA** (Differential Electro Magnetic Analysis) which can be performed with contactless smartcards [5]. By measuring and evaluating the electromagnetic emanation and correlating it with the code running on the chip, information about a secret key stored on it is gathered. A **remote power analysis** was performed by Oren and Shamir [22]. Their attack, targeting at RFID tags operating in the UHF (Ultra High Frequency) range, could probably also be applied to contactless smartcards.

Furthermore, **fault injection**[1] in order to cause a malfunction of the device may reveal a clue to the secret key [2]. A **relay attack** is also feasible [11]: By redirecting the data interchanged between a reader and a tag over a separate communication channel in real time, one can pretend to be the owner of someone else's tag.

The industry wants to keep the prices low and, due to the **restricted energy supply** of the chip via the RF field, the number of switching transistors is limited [19]. Hence, security measures and physical protection on the chip[2] may be very lightweight or won't be employed at all [29], even when security or privacy issues are relevant.

Towards More Security. As fraud involving contactless smartcards is becoming more profitable, soon the first real world offences are expected to emerge. To test and then improve the security of the existing systems, tools being able to perform attacks, as well as to analyse the capabilities and functionality of the used hardware and protocols, need to be developed. As the standards differ very much with regard to operating frequency, communication interface and transmission protocol [9], the hardware for a reasonable security analysis must be custom-made and tailored to a particular one. We opted for the ISO 14443, being the most common and widespread norm for contactless smartcards.

Our Contribution. A cost-efficient embedded system shall be developed to ease the security analysis of, maybe cryptographically enabled, smartcards with an ISO 14443A compliant RF interface. Extensive control of the communication and the energy supply is demanded, as well as interoperability with other hardware and measurement equipment. In addition, stand-alone operation is required for performing practical attacks and mobile data acquisition. Some of the tasks to be made possible are

- communication on the bit layer with a low level reader,
- emulation of an ISO 14443 compliant tag,
- perform practical replay and man-in-the-middle (relay) attacks,
- assist remote power analysis, DEMA and fault injection analysis,
- acquisition and logging of the interchanged data, and
- testing of different types of antennas and power amplifiers.

2 ISO 14443 RFID Operation Principle

As depicted in Fig. 1, a minimum RFID system consists of two main components, namely a reader generating a sinusoidal field with a carrier frequency of $f_c = 13.56\,\text{MHz}$ which supplies the second component of the system, a tag or transponder, with energy and often a clock. Both devices are equipped with a coupling element which in the case of the ISO 14443 is a coil with typical 3-10 windings, allowing for data transfer in both directions.

[1] For instance by manipulating the energy supply or by emission of laser pulses.

[2] Including masking and sensors for detecting fault injection or light.

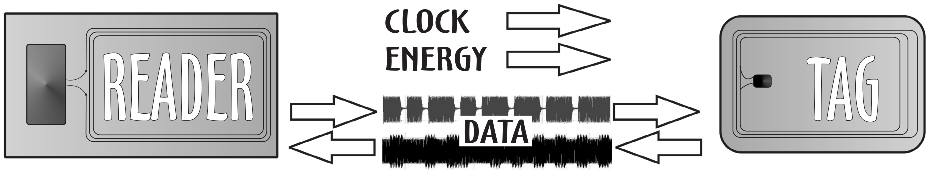


Fig. 1. RFID Operation Principle

The wavelength $\lambda = \frac{c}{f}$ of the electromagnetic field, where c denotes the speed of light and f the carrier frequency, is approximately 22.1 m at 13.56 MHz and therefore several times greater than the typical operating distance of 8-15 cm between reader and tag. Accordingly, the field emitted from the coil[3] of the reader may be treated as purely magnetic[4], leading to the term **inductive coupling** for describing the communication and energy link between reader and tag [9].

Reader → Tag. The reader sends data to the tag using a modified (pulsed) Miller code [9]. Pauses have to be created with a duration of approximately 2.5 μs with 100 % ASK (Amplitude Shift Keying), i.e., the field has to be completely switched off and on by the reader (compare with the upper waveform in Fig. 1).

Tag → Reader. Due to the inductive coupling, the feedback of the transponder drawing more or less energy from the field can be sensed on the side of the reader. Hence, the tag transmits data by switching on and off an additional load and thereby deliberately drawing more energy from the field than during normal operation. This process is termed **load modulation**. As the coupling between tag and reader is pretty weak, the resulting effect on the field is almost not noticeable (compare with the lower waveform in Fig. 1). For this reason, a subcarrier of the frequency of the reader is used for the load modulation, resulting in the transmitted information being placed in sidebands and so making its detection possible [9]. The data is transmitted employing Manchester code and synchronously to the field of the reader, utilising the described OOK (On-Off Keying) with a subcarrier of $\frac{f_c}{16} = 847.5\,\text{kHz}$.

3 Implementation Details of the Embedded System

The developed embedded system consists of a multi purpose reader device which is equipped with a μC (microcontroller), an RF interface and some components for signal processing. A second device, termed *fake tag*, is designed to appear like an authentic tag to an RFID reader and furthermore can acquire the information contained in the field. Between the two units, a communication link can be established.

[3] The technical term for coil is inductivity.

[4] Similar to the common transformer principle.

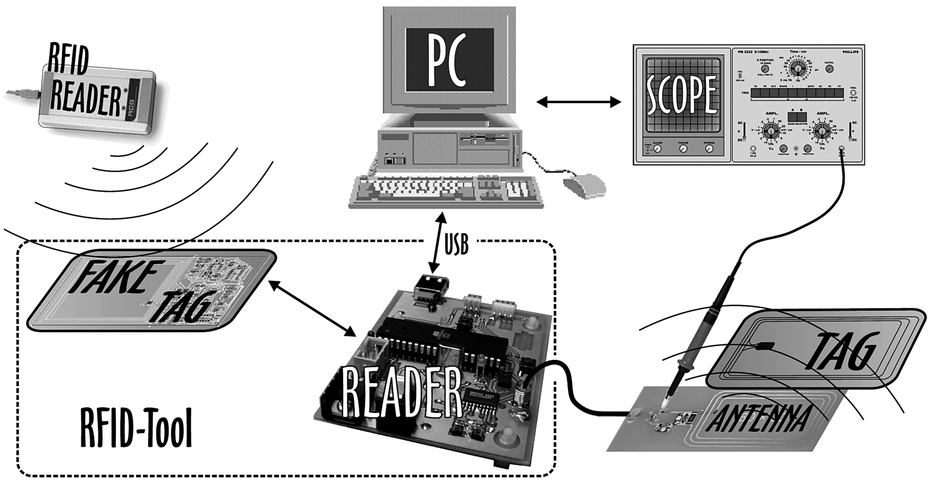


Fig. 2. System Overview

As depicted in Fig. 2, the RFID tool is effortlessly integrated in a measurement setup consisting of a PC (Personal Computer), the developed reader and fake tag, a digital oscilloscope and more equipment for measuring and inducing faults. The PC controls the measurements and later combines and further processes the data acquired from scope and reader. This work focuses on ISO 14443 type A devices using a data rate of 106 $\frac{kBit}{s}$, as specified in the standard [14].

3.1 Reader

The operation principle of the low level reader, as detailed in this section, is depicted in Fig. 3. The RFID tool is based on an Atmel ATMega32 [1] micro-controller, clocked at 13.56 MHz, which is amongst others equipped with 32 kB Flash RAM, 1 kB non-volatile EEPROM(Electronically Erasable Programmable Read Only Memory) and an ADC (Analog to Digital Converter). For flexible operation and testing, the software running on the μC can be updated through a PC without the need to remove it from the PCB (Printed Circuit Board).

The main part of the analogue front end is provided by the EM 4094 **RF-transceiver** [6] which possesses a 200 mW push pull transmitter operating at 13.56 MHz, is capable of 100% ASK and ready for ISO 14443A operation at a price of less than 5 €. The received HF-Signal can be conditioned by internal filters and adjustable receiver gain. The chip allows for transparent operation, i.e., a high input level on its DIN pin will instantly switch off the field, while a low level switches it on, thus enabling flexible, direct control of the RF field. The output stage of the transceiver has been matched for feeding the signal into

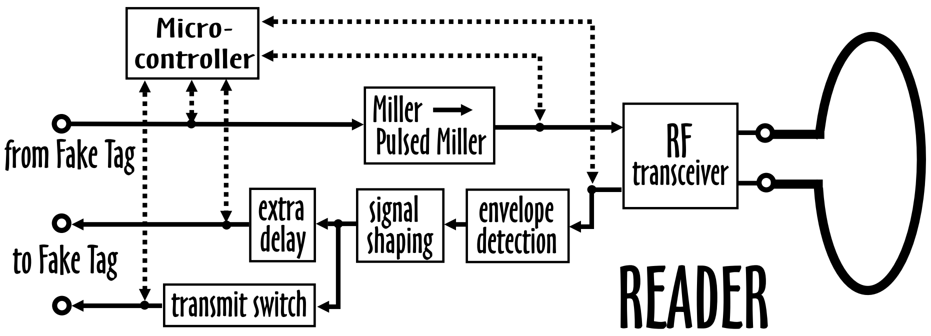


Fig. 3. Operation Principle of the RFID-Reader

a common 50 Ω coaxial cable, so that different antennas and power amplifiers can be connected to a socket placed on the PCB.

Fast communication with a PC or other **USB** (Universal Serial Bus) equipped hardware is made possible by the FT 245R [10] parallel to USB chip from FTDI[5]. The device allows for receiving or sending of packets of eight data bits by pulling a read or write input pin high and low. Using the supplied VCP (Virtual Com Port) driver, a maximum data transfer rate of 1 $\frac{MByte}{s}$ is possible, while the USB port appears as a standard serial COM port, so that a reliable communication can be established fairly easy.

To disburden the μC, the **creation of pauses** (see Sect. 2) is sourced out to a 74123 [26] monoflop, creating the required pulses on every rising edge emitted by the μC. These are fed into the EM 4094 transceiver, resulting in the field being switched off shortly. Two more monoflops, creating pulses on any type of transition, convert Miller coded data, for instance received from the fake tag during a relay attack, to pulsed Miller coded data which is again applied to the field.

The modulated Manchester code, output by the EM 4094, is demodulated using an **envelope detector** circuit. The signal is rectified by a diode and then fed into a LPF (Low Pass Filter). An LM 311 [21] voltage comparator decides whether the subcarrier is present or not, resulting in Manchester encoded data with the appropriate 0 and 5 V levels at its output.

As the demodulation of the signal received from the RF transceiver costs some time (in this case $\approx 1.5\,\mu$s), it can happen, that the answer of the tag is not well synchronised with the reader when relaying data. To take this into account, a circuit has been developed for adding a short **adjustable time delay** to the outgoing signal, without altering its waveform.

An interface for serial communication between the developed reader and fake tag is also installed on the PCB. The data pins can be driven directly by the peripheral circuitry of the RFID tool or steered by the μC, which allows for the emulation of a tag as well as for μC-based processing of the interchanged signals.

[5] www.ftdichip.com

3.2 Fake Tag

The counterpart to the reader, named fake tag, can be utilised for relay attacks as well as for stand-alone emulation of a contactless smartcard. Its functional principle is depicted in Fig. 4. Unlike a normal (passive) tag, the fake tag described here has its own power supply[6] which may also be used for supplying a radio module for communicating with the reader.

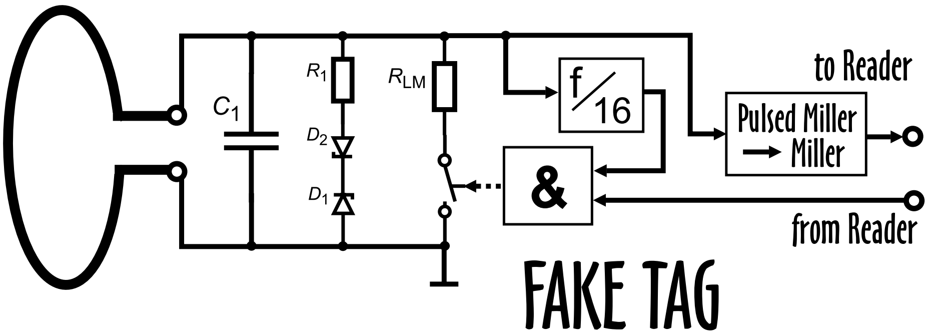


Fig. 4. Operation Principle of the Fake-Tag

A tag needs a coil to establish the coupling with its counterpart at the reader (see Sect. 2). A capacitor is connected in parallel to this inductance, to form a **parallel resonant circuit**. For an ideal parallel resonant circuit, $f_c = \frac{1}{\sqrt{LC}}$ applies [32], where f_c denotes the carrier frequency of the reader, C the capacitance and L the inductance of the tuned circuit. In practice, first the value for L is derived from the shape and dimensions of the coil. Afterwards, the optimal C is calculated and then realised as a variable capacitor, so that the circuit can be tuned more precisely later on. The induced voltage can become relatively large, so two antiparallel Zener-diodes limit the maximum possible voltage and thus protect the rest of the circuit.

The **subcarrier** with a frequency of $\frac{f_c}{16}$ =847.5 kHz is derived from the field generated by the reader. For this, the antenna is connected to the input of a 4-bit binary counter 74393 [23] through a resistor which limits the maximum current into the input stage, as proposed in [7]. The fourth output of the binary counter toggles every $2^3 = 8$ clock cycles which equals frequency division by 16, i.e., the desired subcarrier. For modulating the incoming Manchester coded signal with the subcarrier, a 7408[24] AND gate combines it with the output of the binary counter.

To achieve the **load modulation**, as described in Sect. 2, a resistor has to be connected in parallel to the coil of the tag. This is realised via an IRFD 110 [13] N-channel MOSFET, allowing for fast switching and a maximum drain-source voltage of 100 V. The output of the AND gate (see above) is connected to the gate of the transistor. Accordingly, by toggling the resistor, the 848 kHz-modulated

[6] Can be a small lithium battery.

Manchester code is in turn modulated onto the 13.56 MHz field of the reader and the information put into the sidebands of the carrier.

To **acquire data** from a nearby reader, an LM 311 comparator combined with two envelope detectors (as detailed in Sect. 3.1) are connected in parallel to the resonant circuit. One of the detectors has a fast response time and distinguishes between the field being completely switched off and the load modulation case. The other envelope detector reacts slower and averages the signal at the antenna, for adapting the threshold voltage of the comparator to the current field strength. This approach immunises the circuit to noise caused by the RF field and so extends the operating range.

The output of the comparator is connected to a 7474 D-type flip-flop [25], whose inverted output is fed back into its input. Hence, a change of the output state occurs on every rising edge at the input. This conversion from pulses into transitions, resulting in a Miller coded data signal, is amongst others necessary to reduce the bandwidth required for the communication link of the RFID tool.

3.3 Operation Modes

The software for the μC is mainly written in C, with assembler code inserted, where the execution speed is crucial. Besides, a C library for controlling the RFID tool from a PC, as well as a corresponding GUI (Graphical User Interface) is available. The following operating modes are currently implemented:

- *bit level reader*: the reader is freely controlled by the PC via USB,
- *stand-alone reader*: mobile operation with an arbitrary command sequence prestored in the EEPROM (and acquired data stored into it),
- *tag emulation*: the fake tag is directly controlled via USB,
- *mobile tag emulation*: prestored data is replayed by the fake tag, while the reader's requests are recorded to the EEPROM,
- *relay mode*: mobile operation of both reader and fake tag, while the relayed bits in both directions can be recorded to the EEPROM.

Further routines are provided for generating ISO 14443 compliant bitstreams and for reading and writing the non-volatile EEPROM.

4 Results

The flexible **low level reader** mode has been successfully tested with several ISO 14443 compliant tags which are partly listed below in this section. The exact behavior and timing of the contactless interface can be flexibly steered, even transcending the ISO standard, if desired.

The data sent out by the **fake tag** is accepted by an ACG[7] Dual 2.1 Passport Reader in our laboratory, just as if it was a genuine tag. During our tests, the answer of the fake tag to a request issued by the reader was intentionally delayed

[7] http://acg-id.aaitg.com

by more than 250 μs and the resulting behaviour was analysed. Compliance of the ACG reader with the strict timing requirements during the initialisation phase[8] could not be observed, i.e., the delayed answer was still accepted, thus easing relay attacks.

The RFID tool can be used for **logging the data** interchanged in any direction. This can be helpful to analyse unknown protocols, as well as for further processing, e.g., key-search with cost effective hardware, such as proposed in [18].

Various **antennas** were built, tuned to resonance with the carrier frequency and matched to a 50 Ω coaxial cable, to perform tests with regard to the operating range and the influence of the physical environment of the card.

For executing a **relay attack** [16], the antenna of the bit level reader possessed by the offender has to be placed close enough to a contactless card of the victim. At the same time, the fake tag is brought into the field of an RFID reader, e.g., at the cash desk, by an accomplice. The data being transferred by this reader is acquired and directly forwarded on the bit layer through a communication link to the attacker. There, the data is retransmitted to the card of the victim. Its answer is relayed back to the reader at the cashpoint and so, as the attackers continue relaying the data, both reader and tag will be convinced that they are in close vicinity to each other and thus carry out their task, e.g., authorise a payment.

Such an attack has been successfully carried out using the here described embedded tool with

– an RFID-enabled passport, issued by the Federal Republic of Germany,
– a student identity chip card of the Ruhr-University in Bochum, Germany,
– Philips classic Mifare and DESFire cryptographically enabled smartcards,
– an Atmel AT88SC153 smartcard, and
– a ticket for the FIFA world cup 2006 in Germany,

until to at least reading out the UID (Unique Identifier) of the tags. In the case of the Mifare classic, after a the successful login, encrypted data blocks were read out and modified remotely. Furthermore, the 64 Byte content of a world cup ticket was read out using the relay mode and the interchanged data was recorded for subsequently analysing the protocol. The Philips Mifare Ultralight chip embedded in the ticket [28] provides no encryption at all. Hence, the RFID access control could easily be spoofed with the developed embedded system, by means of a replay attack, as the communication protocol is fully published in the data sheet [27].

When relaying data, a delay is inevitable, as described in Sect. 3.1. However, if a reader scrutinised the timing, a relay attack could still be carried out successfully, as the (fixed) bit sequence of a command could be stored in the μC and sent out instantly after an incoming request.

Hancke and Kuhn [12] proposed a countermeasure for relay attacks, based on ultra-wideband pulses. Still, as it is not employed in current tags, the most effective way to enhance privacy is constructing a Faraday's cage for the tag: Our experiments proved, that a single layer of aluminum foil wrapped around

[8] ISO 14443 requires the tag to answer to a *REQA* exactly after 86.9 μs.

the smartcard completely protects it from being activated or read out by an unauthorised reader.

The implemented embedded system has become a valuable part of the measurement setup in our laboratory and is currently employed to assist several ongoing security analyses (compare with Sect. 5).

5 Future Prospects

At the moment, the achieved **read range** with the developed reader and the antennas used is approximately 5-10 cm. It is possible to extend this range to 25 cm [17], using a power amplifier [20] and a large copper tube antenna [30].

The communication protocol of a Philips Mifare DESFire contactless smartcard has been reverse engineered until to the point necessary for carrying out a **DEMA** [4]. In the respective attack, the challenges[9] were generated by a proprietary RFID reader and had to be extracted from the oscilloscope waveforms, which meant a severe, time consuming constraint for the analysis. Using the developed system, arbitrary access to the contactless interface is provided, allowing amongst others for freely chosen challenges. A DEMA is based on a statistical test at one certain point in time, so subsequent measurements need to be synchronised before superimposing them. The for his purpose required reliable signal to trigger the oscilloscope can now also be emitted by the RFID tool, thus further improving the attack.

It is promising to use the embedded system for execution of a **remote power analysis**. During the pauses occurring in the field of the reader (compare with Sect. 2), a tag draws its energy from a built-in capacitor which recharges, when the field is activated again. Consequently, different shaped energy peaks emerge in the field, depending on the amount of power consumed by the tag during the energy gap. This behaviour might be exploited to derive a secret key stored on the tag. The RFID tool provides a corresponding output signal which can be acquired by the Atmel's internal ADC or an oscilloscope.

As the reader can be arbitrarily programmed, **fault injection attacks** are feasible [2] in which the device is forced to show erroneous performance, by perturbing physical parameters like the power supply or the clock frequency. Furthermore, controlling of external pulse generators and other fault injection equipment with the RFID tool is possible.

Finally, any **new protocols** based on the ISO 14443 standard can be implemented and tested. If additional hardware was required, it could easily be connected to the PCB.

6 Conclusion

In this contribution, we present an embedded implementation of a cost effective, arbitrarily programmable RFID reader and a fake tag which can be used for various promising purposes. The tool was built using electronic hobbyist equipment

[9] Random numbers interchanged for the authentification.

and off the shelf components at a cost of less than 40 €, and its design is simple enough to be reproduced by a low skilled attacker. With the developed hardware, we have successfully carried out relay and replay attacks between various contactless smartcards and a commercial RFID reader. Integrated in a measurement system, the proposed tool can help to carry out security analyses, such as a DEMA or a remote power analysis, and assist fault injection attacks. The stand-alone operation modes permit for mobile tag emulation, reader operation and logging of the interchanged data.

Employing ISO 14443 compliant contactless smartcards in security sensitive applications should be regarded very critically, as the physical interface is proven to be insecure against relay attacks. A smartcard identified by a reader does not have to be in its direct vicinity, as declared by many manufacturers. Instead, the data can be forwarded from large distances without permission or even notification of the owner, as described in this paper, with little effort. If an RFID tag is indispensable, we suggest a metal shielding to prevent unauthorised access and propose that the card should not be able to become active, unless the owner has performed an action, e.g., press a button or open the cover of his electronic passport.

References

1. Atmel. ATMega32 data sheet. http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf.
2. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. *Lecture Notes in Computer Science*, 1294:513–, 1997.
3. BSI - German Ministry of Security. ePass - Der Reisepass mit biometrischen Merkmalen. http://www.bsi.de/fachthem/epass/.
4. D. Carluccio. Electromagnetic Side Channel Analysis for Embedded Crypto Devices. Master's thesis, Chair for Communication Security at the Ruhr University Bochum, 2005. Diploma thesis.
5. D. Carluccio, K. Lemke, and C. Paar. Electromagnetic side channel analysis of a contactless smart card: first results. In *ECRYPT Workshop on RFID and Lightweight Crypto*, pages 44–51, Graz, Austria, July 2005. ECRYPT. http://www.iaik.tu-graz.ac.at/research/krypto/events/RFID-Slidesand Proceedings/Proceedings-WSonRFIDandLWCrypto.zip.
6. EM Microelectronic. EM4094 fact sheet. http://www.emmicroelectronics.com/webfiles/product/rfid/ds/EM4094_fs.pdf
7. Fairchild Semiconductors. Application note 313: DC electrical characteristics of MM74HC high speed logic. http://www.fairchildsemi.com/an/AN/AN-313.pdf.
8. T. Finke and H. Kelter. Radio Frequency Identification Abhörmöglichkeiten der Kommunikation zwischen Lesegerät und Transponder am Beispiel eines ISO14443-Systems. http://www.bsi.de/fachthem/rfid/Abh_RFID.pdf. BSI - German Ministry of Security.
9. K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley and Sons, 2nd edition, 2003.
10. FTDI. FT245 USB chip data sheet. http://www.ftdichip.com/Documents/DataSheets/DS_FT245R_v105.pdf.
11. G. Hancke. A practical relay attack on ISO 14443 proximity cards. http://www.cl.cam.ac.uk/ gh275/relay.pdf, 2005.

12. G. P. Hancke and M. G. Kuhn. An RFID distance bounding protocol. In *Proceedings of IEEE/Create-Net SecureComm 2005*, pages 67–73. IEEE Computer Society Press, 2005.
13. International Rectifier. Data sheet for IRFD110 N-channel MOSFET. http://www.irf.com/product-info/datasheets/data/irfd110.pdf.
14. ISO/IEC 14443. Identification cards - Contactless integrated circuit(s) cards - Proximity cards - part 1-4. www.iso.ch, 2001.
15. A. Juels, D. Molnar, and D. Wagner. Security and privacy issues in e-passports. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005.*, pages 74–88. IEEE, September 2005.
16. Z. Kfir and A. Wool. Picking virtual pockets using relay attacks on contactless smartcard systems. Cryptology ePrint Archive, Report 2005/052, 2005. http://eprint.iacr.org.
17. I. Kirschenbaum and A. Wool. How to build a low-cost, extended-range RFID skimmer. Cryptology ePrint Archive, Report 2006/054, 2006. http://eprint.iacr.org/.
18. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler. How to break DES for 8,980. In *International Workshop on Special-Purpose Hardware for Attacking Cryptographic Systems — SHARCS'06, Cologne, Germany*, April 2006.
19. T. Lohmann, M. Schneider, and C. Ruland. Analysis of power constraints for cryptographic algorithms in mid-cost RFID tags. In J. Domingo-Ferrer, J. Posegga, and D. Schreckling, editors, *Smart Card Research and Advanced Applications*, volume 3928 of *Lecture Notes in Computer Science*, pages 278–288. Springer, 2006.
20. Melexis. Application note: A power booster for the MLX90121. http://www.melexis.com/prodfiles/0003881_AN90121_4_1.pdf.
21. National Semiconductor. Datasheet for LM311 voltage comparator. http://www.national.com/pf/LM/LM311.html#Datasheet.
22. Y. Oren and A. Shamir. Power analysis of RFID tags. http://www.wisdom.weizmann.ac.il/ yossio/rfid/.
23. Philips. Data sheet for 4 bit binary ripple counter 74393. http://www.semiconductors.philips.com/pip/74HC393D#datasheet.
24. Philips. Data sheet for 7408 AND gate. http://www.semiconductors.philips.com/pip/74HC08N.
25. Philips. Data sheet for D type flip-flop 7474. http://www.semiconductors.philips.com/pip/74F74.html#datasheet.
26. Philips. Data sheet for monostable multivibrator 74HC/HCT123. http://www.semiconductors.philips.com/pip/74HCT123D#datasheet.
27. Philips. Data sheet for MIFARE Ultralight Contactless Single-trip Ticket IC. http://www.semiconductors.philips.com, 2003.
28. Philips. Philips scores in German stadiums. *On the move*, page 3, Mar 2006.
29. M. R. Rieback, B. Crispo, and A. S. Tanenbaum. The evolution of RFID security. *Pervasive Computing*, 5(1), Jan-Mar 2006.
30. Texas Instruments. HF Antenna Cookbook Technical Application Report. http://www.ti.com/rfid/docs/manuals/appNotes/HFAntennaCookbook.pdf, 2004.
31. Texas Intruments. Texas Instruments to deliver RFID solution for MasterCard PayPass. http://www.ti.com/rfid/docs/news/news_releases/2005/ rel01-17-05a.shtml.
32. U. Tietze and C. Schenk. *Halbleiter-Schaltungstechnik*. Springer, eleventh edition, 2001.

A Comparative Analysis of Common Threats, Vulnerabilities, Attacks and Countermeasures Within Smart Card and Wireless Sensor Network Node Technologies

Kevin Eagles, Konstantinos Markantonakis, and Keith Mayes

Smart Card Centre
Information Security Group
Royal Holloway, University of London,
Egham, England TW20 0EX
k.eagles@sensornets.co.uk,
{k.eagles,k.markantonakis,keith.mayes}@rhul.ac.uk

Abstract. A threat analysis framework and methodology was developed by the authors to catalogue threats, vulnerabilities, attacks and countermeasures for smart cards (contact and contactless) and wireless sensor network node technologies. The goal of this research was to determine "Security Lessons" learned from the world of smart cards that may be applied to wireless sensor network nodes and vice versa.

Keywords: Sensor Networks, Security, Ubiquitous Networks, RFID, Smart Card Security, Security Models.

1 Introduction

Smart cards and wireless sensor network nodes (hereafter referred to as WSN nodes) are two functionally distinct technologies sharing similar design characteristics. Both have severe space and computational restrictions and require low levels of power to function.

Smart cards have evolved from being simple insecure data carriers, to quite sophisticated devices today (e.g., mobile cell phone SIM technology). There are many standards that govern the development and use of smart cards and many vendors in the market place.

Conversely, WSN nodes are a relatively new form of evolving technology and although products are widely available, there are only a few embryonic standards governing development and use [1].

With ever increasing miniaturisation and ubiquity of computing devices there may be overlapping areas within technologies such as smart cards and WSN nodes (and indeed PDAs, laptops and mobile cell phone technology too). The proposed framework and methodology for the systematic analysis of security issues within this paper may help to assess potential overlaps and/or convergences within technologies.

To enable this work, two high level objectives were established.

- OBJECTIVE 1: Determine if there are any security threats, vulnerabilities, attacks and countermeasures that have been established for smart card technologies (both contact and contactless) that can be directly and/or indirectly applied to wireless sensor network node technologies;
- OBJECTIVE 2: Determine if there are any existing or emergent security threats, vulnerabilities, attacks and countermeasures that have been established for wireless sensor network node technologies that can be directly and/or indirectly applied to smart card technologies.

The rest of this paper has the following structure: Section two outlines threat and attack models for smart cards. Section three outlines threat and attack models for WSN nodes. Section four discusses the framework and methodology established for the data capture phase of the study. Section 5 discusses the comparative threat analysis of the two technologies. Finally, section six provides conclusions and recommendations for further work.

2 Smart Card Threat and Attack Models

Historically, smart cards have endured many threats and attacks exposing vulnerabilities, however most of these threats now have effective countermeasures. Many smart cards have not been through a recognised security evaluation, however, it is important to note that some industries (e.g., banking/credit cards) have insisted that certain aspects of smart card technology are assessed through Common Criteria [2]. We believe that historically the drive to seek Common Criteria [2] evaluations has helped firm and mature security requirements and functionalities within many technologies.

2.1 Smart Card Definitions

A smart card consists of an integrated circuit with some form of tamper resistance, packaged and embedded within a card carrier. Overarching definitions of smart card technologies follow:

"The integrated circuit is a single chip incorporating CPU and memory which may include RAM, ROM, and/or programmable non-volatile memory (typically EEPROM or Flash Memory) [3]."

"The chip is embedded in a module which provides the capability for standardised connection to systems separate from the chip [3]."

The card carrier is usually made from plastic and typically conforms to ISO/IEC 7810:2003 [4] and ISO/IEC 7813:2006 [5].

Smart cards can be broken down into contact and contactless varieties.

Contact cards are typically in accordance with the standard ISO/IEC 7816 (parts 1-15) [6], which covers physical characteristics of the integrated circuit and also the electrical interface and connectivity for both power and data via a card reader.

Contactless cards can be broken down into two main areas, proximity cards and vicinity cards. Proximity cards are typically in accordance with ISO/IEC 14443-1:2000 (parts 1 to 4) [7]. Power and data are transferred via inductive coupling over a distance not exceeding 10cm. Vicinity cards are typically in accordance with ISO/IEC 15693 (parts 1 to 3) [8]. Power and data are transferred via inductive coupling over a distance not exceeding 1.2 metres.

Within this paper, the term smart card will be used to cover both contact and contactless smart cards (unless a distinction needs to be made).

2.2 Radio Frequency (RF) and Radio Frequency Identification (RFID) Definitions

Contactless smart cards utilise radio frequency fields for their communications and usually (although not always) as a source of power.

RFID devices are not restricted to card carriers and can be embedded into a range of objects, they are less sophisticated than contactless smart cards due to functional and cost requirements rather than technical limitations.

"RFID refers to procedures to automatically identify objects using radio waves [9]".

2.3 Smart Card Threats

We have derived Threat/Attack groups from the following research [10, 11]. These groups map effectively to popular generic attacker groupings

- Class I (Clever outsiders): Smart but lack sufficient knowledge of the system having access to only moderately sophisticated equipment. They exploit existing weaknesses rather than creating any. "Opportunist Attacker" (Hobbyist/Vandal/Minor Criminal possibly using widely available tools);
- Class II (Knowledgeable insiders): Substantial and specialised technical education and experience, understanding parts of the system and potential access to most of it. They have sophisticated tools and instruments for analysis. "Expert/Professional Attacker/Major Criminal" (Personal gain generally financially motivated, using tools adapted specifically for the purpose);
- Class III (Funded organisations): Specialist teams with related and complementary skills backed up with significant resources. Capable of in-depth analysis of the system, designing sophisticated attacks, and have the most advanced analysis tools available. They may use Class II adversaries as part of the attack team. "Sophisticated Attacker" (Intelligence Services, highly resourced Research Labs or very highly skilled Organised Crime).

2.4 Smart Card Attacks

This subsection will outline well known and established attacks on smart cards. Often an attacker is aiming to 'Reverse Engineer' the technology to establish how it works [10, 12]. The main objective is to identify the structure of the chip as well as detailed information on its internal operations.

Invasive Attacks. To gain unauthorised disclosure or modification of security features/functions, user data, software operation, other operational information and/or change the behaviour of the chip. This is done by physical probing and/or physical modification of the chip.

Semi-Invasive Attacks. Skorobogatov [12] describes semi-invasive attacks involving some depackaging to reach the chip's surface, however it is not necessary to break through the passivation layer to access the chip's interior (e.g., the use of light to induce a processing fault).

Non-Invasive Attacks. This type of attack is aimed at retrieving sensitive data (e.g., keys) while observing a smart card under operation or stress. Leakage may occur through emanations, variations in power consumption, Input/Output characteristics, clock frequency, or by changes in processing time requirements.

Observation Attacks: Information Leakage and/or Cryptanalysis. Kocher [13] described an attack on the RSA algorithm by conducting timing attacks on a CPU to count and log cycles between known events (e.g., measure the decryption times for several known cipher-texts) in order to obtain decryption keys.

Simple Power Analysis (SPA) is an analysis of power consumption to determine which set of CPU instructions are being processed and under which parameters. Differential Power Analysis (DPA) is similar to SPA but differs due to the measurement of power when known data is processed by the card and results are statistically analysed to look for patterns. Differential Electro-Magnetic (Radiation) Analysis (DEMA) looks at the electromagnetic emanation from the smart card to attempt to retrieve sensitive data. Differential Fault Analysis (DFA) aims to retrieve secret information from the card by inducing an error whilst a cryptographic calculation is being performed by the card. With the exception of DFA, these attacks are sometimes known as Side Channel attacks.

Protocol and/or Functionality attacks. This type of attack looks for flaws in the protocol implementation. Techniques can be replay-attacks or interrupting the smart card while it is executing a command.

Software Attacks. This type of attack is looking into software malfunctions of the smart card (e.g., software loading and badly formatted commands aiming to circumvent security mechanisms on the card).

Deficiency of Random Numbers. An attacker may predict or obtain information about random numbers generated by the microcontroller because of poor quality entropy and/or seeding of the random numbers created.

Perturbation, Malfunction, State, Environmental Stress. This involves operating the smart card outside of its normal operating conditions (e.g., increasing or decreasing operational temperatures) to attempt to deactivate security features or disclose information.

3 WSN Node Threat and Attack Models

Although there is research on Java Card 3.0 [14] and TCP/IP and there has been research with secure distributed computing on a Java Card grid [15], the typical usage of smart cards today is not as networked devices; conversely a WSN node is a networked device.

3.1 Wireless Sensor Network Nodes Definitions

The term ‘Mote’ (originally labelled COTS Dust [16]) is often interchangeable with the notion of a sensor node or wireless network node. For this paper, a WSN node refers to a device consisting of an integrated circuit with a microprocessor and memory which is able to function as an element within a network, passing data onto other devices through wireless communications.

“These devices make up hundreds or thousands of ad hoc tiny sensor nodes spread across a geographical area. These sensor nodes collaborate among themselves to establish a sensing network. A sensor network that can provide access to information anytime, anywhere by collecting, processing, analysing and disseminating data [17]”.

3.2 WSN Node Threats

Initially, many WSN routing protocols were vulnerable to targeted attacks, to some degree this is still the case. Although there are many ‘open’ routing protocols today, some implementations use proprietary routing protocols and algorithms.

Many papers categorise threats as being network Outsiders or Insiders [18, 19, 20]; further, attackers are categorised as Mote-class attackers or laptop-class attackers [18, 19]. Mote-class attackers are perceived to have access to only a few WSN nodes to exploit and derive weaknesses; they also have an attack surface affecting only a few nodes within a WSN. Conversely, a laptop-class attacker may be in possession of much more potent devices (e.g., laptops for instance).

3.3 WSN Node Attacks

WSN nodes have limited storage, processing and bandwidth capability and power (battery) management is essential [17].

C. Karlof and D. Wagner 2003 [18] state “Insider attacks may be mounted from either compromised sensor nodes running malicious code or adversaries who have stolen the key material, code, and data from legitimate nodes, and who then use one or more laptop-class devices to attack the network.”

Attacks tend to focus on the nature of WSN nodes and known vulnerabilities:

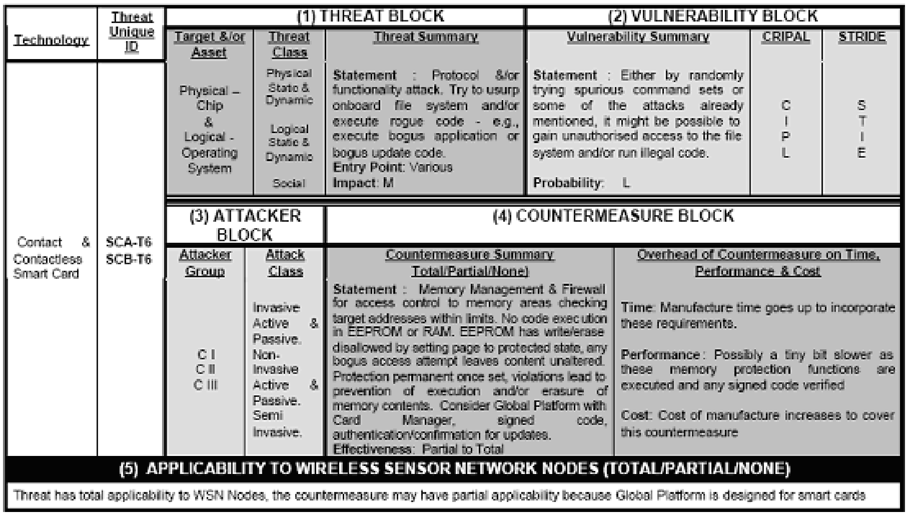
- Denial of Service attacks on the device by running down the power source (battery) through continuous operation;
- Denial of Service through Radio Frequency jamming so data can not be transmitted or received;

– Most (if not all) devices do not seem to have a crypto-coprocessor, thus any encryption creates a processing overhead for already constrained capabilities;
– WSN node Integrated Circuits are not tamper resistant, any secret information on the chip may be susceptible to standard smart card attacks.

We were not able to find any references for WSN node Common Criteria [2] evaluated products or Protection Profiles to enable evaluations. However, NIST are involved with the US Department of Homeland Security in the development of advanced CBRNE - (chemical, biological, radiological, nuclear, and explosive) detection sensors that could provide the underpinnings for a national sensor network [21, 22].

4 Threat, Vulnerability, Attacker and Countermeasure Table

To capture and categorise data, we created a framework and methodology in the form of a TVAC Table (Fig. 1).



Technology	Threat Unique ID	(1) THREAT BLOCK			(2) VULNERABILITY BLOCK		
		Target &/or Asset	Threat Class	Threat Summary	Vulnerability Summary	CRIPAL	STRIDE
Contact & Contactless Smart Card	SCA-T6 SCB-T6	Physical – Chip & Logical - Operating System	Physical Static & Dynamic Logical Static & Dynamic Social	**Statement** : Protocol &/or functionality attack. Try to usurp onboard file system and/or execute rogue code - e.g., execute bogus application or bogus update code. **Entry Point**: Various **Impact**: M	**Statement** : Either by randomly trying spurious command sets or some of the attacks already mentioned, it might be possible to gain unauthorised access to the file system and/or run illegal code. **Probability**: L	C I P L	S T I E

Technology	Threat Unique ID	(3) ATTACKER BLOCK		(4) COUNTERMEASURE BLOCK	
		Attacker Group	Attack Class	Countermeasure Summary Total/Partial/None)	Overhead of Countermeasure on Time, Performance & Cost
Contact & Contactless Smart Card	SCA-T6 SCB-T6	C I C II C III	Invasive Active & Passive. Non-Invasive Active & Passive. Semi Invasive.	**Statement** : Memory Management & Firewall for access control to memory areas checking target addresses within limits. No code execution in EEPROM or RAM. EEPROM has write/erase disallowed by setting page to protected state, any bogus access attempt leaves content unaltered. Protection permanent once set, violations lead to prevention of execution and/or erasure of memory contents. Consider Global Platform with Card Manager, signed code, authentication/confirmation for updates. **Effectiveness**: Partial to Total	**Time**: Manufacture time goes up to incorporate these requirements. **Performance**: Possibly a tiny bit slower as these memory protection functions are executed and any signed code verified **Cost**: Cost of manufacture increases to cover this countermeasure

(5) APPLICABILITY TO WIRELESS SENSOR NETWORK NODES (TOTAL/PARTIAL/NONE)
Threat has total applicability to WSN Nodes, the countermeasure may have partial applicability because Global Platform is designed for smart cards

Fig. 1. A Sample Threat, Vulnerability, Attack and Countermeasure (TVAC) Table

The TVAC table has five main blocks with subsections. It has two initial columns categorising the technology and its unique identifier (TUID). A contact smart card is prefixed SCA, a contactless smart card prefixed SCB and a WSN node prefixed WSNN.

4.1 Threat Block

In the context of this project, we have defined a threat as being, “an objective a foe might try to realise in order to misuse a target or asset.”

Target and/or Asset: The following categories are used to categorise the threat type:

- Physical - Chip; Physical - Other;
- Logical - Operating System; Logical - Platform;
- Logical - Application; Logical - Other;
- Communications Bearer (e.g., Physical Card Reader, RF or RFID);
- Other.

Threat Class: The classification of the threat type as follows:

- Physical Static (e.g., No Power to Hardware);
- Physical Dynamic (e.g., Power to Hardware);
- Logical Static (e.g., No Power source active, but using glitches, light, temperature variances to affect software before power activated);
- Logical Dynamic (e.g., Power to Software);
- Social (e.g., Social Engineering);
- Policy (e.g., Weakness in Governing Policies);
- Other.

Threat Summary: This includes a ‘Statement’ of the Threat indicating ‘Entry Point’ and rating the ‘Impact’ of the Threat from High, Moderate or Low.

4.2 Vulnerability Block

In the context of this project, we have defined a vulnerability as being, “a specific means by which a threat can be executed via an unmitigated attack path.”

Vulnerability Summary: A ‘Statement’ of the Vulnerability, with a ‘Probability’ rating from High, Moderate or Low.

CRIPAL: is an acronym and methodology we have established to cover high level ‘primary’ security goals. The acronym stands for: (C)onfidentiality; (R)eliability; (I)ntegrity; (P)rivacy; (A)vailability; (L)egitimate Use.

STRIDE: a method used by Microsoft [23] to categorise threats during software development. This adds low level granularity to ‘CRIPAL’ area. It stands for: (S)poofing, (T)ampering, (R)epudiation, (I)nformation disclosure, (D)enial of Service, (E)levation of Privilege.

4.3 Attacker Block

In the context of this project, we have defined an attacker as being, “the entity that is exploiting a vulnerability to establish a threat.”

Attacker Group: As stated earlier they consist of:

– Class I (clever outsiders) - "Opportunist Attacker";
– Class II (knowledgeable insiders) - "Expert/Professional Attacker";
– Class III (funded organisations) - "Sophisticated Attacker".

Attack Class: This consists of:

– Invasive Active (e.g., Cutting new tracks);
– Invasive Passive (e.g., Microprobing to observe not to modify);
– Non-Invasive Active (e.g., Power Surge or glitch attacks);
– Non-Invasive Passive (e.g., DPA and Timing Attacks);
– Semi Invasive techniques (e.g., Light attacks).

4.4 Countermeasure Block

In the context of this project, we have defined a countermeasure as being, "a mitigation measure that prevents, detects or significantly reduces a misdeed associated with a specific threat or group of threats."

Countermeasure Summary: A 'Statement' of the Countermeasure, indicating its 'Effectiveness' represented by the following options: Total (Complete Effectiveness); Partial (Some Effectiveness); None

Overhead of Countermeasure on Time, Performance & Cost: This looks at what impact the countermeasure may bring if implemented.

4.5 Applicability to WSN Nodes/Smart Cards

This is an assessment of whether any security issues and mitigation can be applied from one technology type to the other represented by the following options: Total (Complete Effectiveness); Partial (Some Effectiveness); or None

5 Comparative Threat Analysis Assessment Matrices

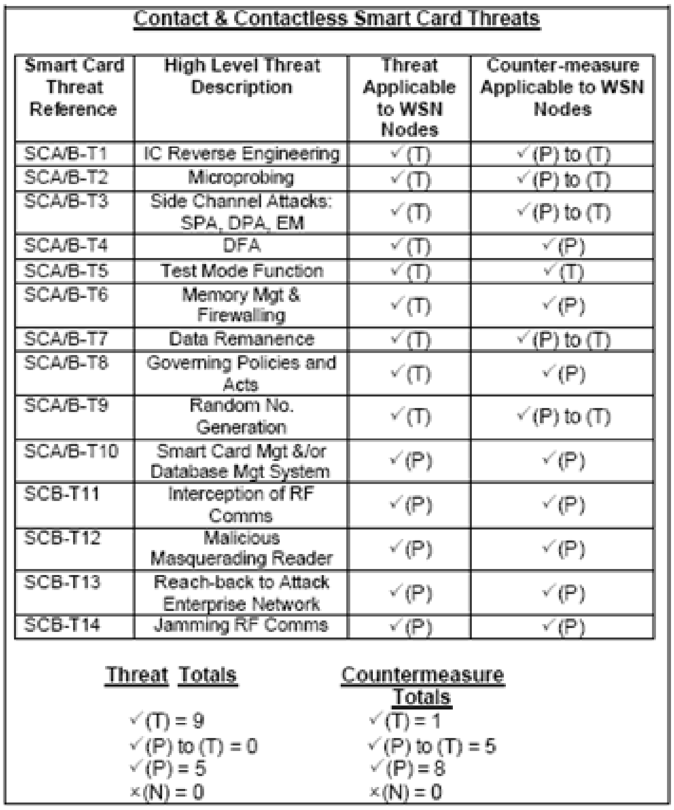
Fig. 2 and Fig. 3 below illustrate Comparative Threat Analysis Assessment Matrices designed by the authors, which use data from populated TVAC tables.

These matrices record any commonality/applicability from one technology to the other. Ten threats, SCA-T1 to SCA-T10, have been explored for contact smart cards and these have also been applicable to contactless smart cards too as SCB-T1 to SCB-T10 respectively. Four additional threats have been applied to contactless smart cards as SCB-T11 to SCB-T14, giving contactless smart cards a count of fourteen. Eight threats were listed for WSN nodes (WSNN-T1 to WSNN-T8).

Key to Matrices:

– SCA/B: Threat and/or Countermeasure is applicable to both Contact and Contactless cards and hence are referenced as so;
– Contact Smart Card: has SCA prefix with threat reference following - e.g., SCA-T1;

- Contactless Smart Card: has SCB prefix with threat reference following - e.g., SCB-T1;
- WSN Node: has WSNN prefix with threat reference following - e.g., WSNN-T1;
- $\surd(T) = TotalMatch; \surd(P)to(T) = PartialtoTotalMatch;$ $\surd(P) = PartialMatch; \times(N) = NoMatch.$



Contact & Contactless Smart Card Threats

Smart Card Threat Reference	High Level Threat Description	Threat Applicable to WSN Nodes	Counter-measure Applicable to WSN Nodes
SCA/B-T1	IC Reverse Engineering	✓(T)	✓(P) to (T)
SCA/B-T2	Microprobing	✓(T)	✓(P) to (T)
SCA/B-T3	Side Channel Attacks: SPA, DPA, EM	✓(T)	✓(P) to (T)
SCA/B-T4	DFA	✓(T)	✓(P)
SCA/B-T5	Test Mode Function	✓(T)	✓(T)
SCA/B-T6	Memory Mgt & Firewalling	✓(T)	✓(P)
SCA/B-T7	Data Remanence	✓(T)	✓(P) to (T)
SCA/B-T8	Governing Policies and Acts	✓(T)	✓(P)
SCA/B-T9	Random No. Generation	✓(T)	✓(P) to (T)
SCA/B-T10	Smart Card Mgt &/or Database Mgt System	✓(P)	✓(P)
SCB-T11	Interception of RF Comms	✓(P)	✓(P)
SCB-T12	Malicious Masquerading Reader	✓(P)	✓(P)
SCB-T13	Reach-back to Attack Enterprise Network	✓(P)	✓(P)
SCB-T14	Jamming RF Comms	✓(P)	✓(P)

Threat Totals	Countermeasure Totals
✓(T) = 9	✓(T) = 1
✓(P) to (T) = 0	✓(P) to (T) = 5
✓(P) = 5	✓(P) = 8
×(N) = 0	×(N) = 0

Fig. 2. Smart Cards Compared to WSN Nodes Matrix

The following is a summarised breakdown of our findings:

SCA/B-T1: Smart cards are susceptible to reverse engineering of the Integrated Circuit (IC). Possible countermeasures include an active shield, mesh or sensor that once affected renders the IC unusable, destroying data on the chip and then shutting down operations. The use of environmental sensors within the chip would have a similar affect. Most smart cards have tamper resistance, most if not all WSN nodes do not. Although some WSN nodes are ruggedised for use in harsh environments, this offers no protection from known threats. Smart card tamper resistance techniques should be transferable to WSN nodes.

SCA/B-T2: Smart cards are susceptible to Microprobing. This attack and its countermeasure are closely related to SCA/B-T1 above.

SCA/B-T3 and SCA/B-T4: Side Channel attacks like SPA, DPA or EM analysis may apply to WSN nodes. Randomness and scrambling countermeasures used within smart cards maybe transferable to WSN nodes. The same stands for DFA which is SCA/B-T4.

SCA/B-T5: This maps to WSNN-T8 and involves a Test Mode which smart card and WSN node chips have [24]. It is possible to unlock the Test Mode function and as such get full logical control of the IC. Smart cards mitigate this by requesting authentication to the Test Mode function with a failure leading to chip inoperability.

SCA/B-T6: Some smart cards undertake a form of internal firewalling with memory management to prevent a protocol or functionality attack. WSN nodes do not have this protection, but could learn from smart card countermeasures.

*SCA/B-T7:*Skorobogatov has undertaken research in the field of Data Remanence. The countermeasures he proposes [12] for the protection of smart cards should be applicable to WSN nodes.

SCA/B-T8: Policies. There is a need for clear operating policies such as CONOPS, CONUSE and CONEMP, and adherence to 'laws of the land' (e.g., UK Data Protection Act 1998). Many publications also mention asymmetric keys within smart cards and WSN nodes and public key certificates; however these publications do not mention a Certificate Policy or Key Management Policy which would underpin the use of keys or certificates.

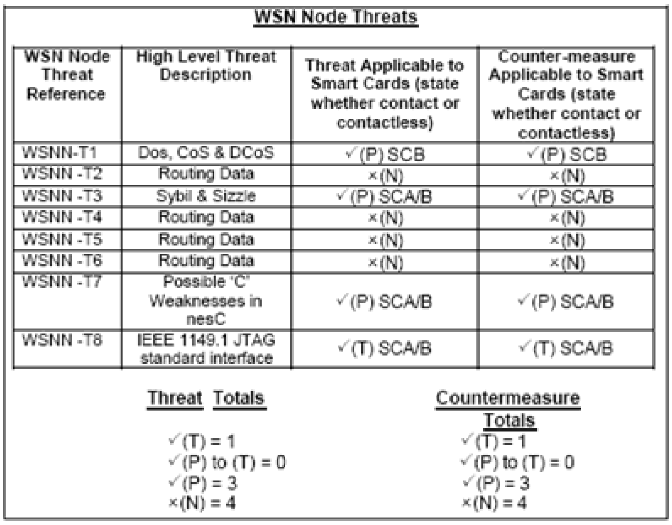
SCA/B-T9: This threat involves weakness in random number generation and many smart cards mitigate this through crypto-coprocessors. WSN nodes do not appear to have crypto-coprocessors and their addition may help with processing capability.

SCA/B-T10: This relates to a Smart Card Management System and/or a Database Management System. These are required for effective management of smart cards but also provide a path for a reach-back attack from a device like a smart card into an Enterprise network. This may apply to WSN nodes, but we have seen no mention of a WSN Node Management System -which in itself seems vulnerability.

SCB-T11 and SCB-T12: These threats involve the interception of messages via RF communications and have partial applicability to WSN nodes. SCB-T11 is an eavesdropping threat between reader & transponder [25]. SCB-T12 is similar but involves a malicious masquerading reader. The countermeasures in both cases are not totally effectively for WSN nodes.

SCB-T13: Potential RFID attacks with SQL, buffer overrun and threat of reach-back to Enterprise networks [26]. A range of RF/RFID exploits may be applied from smart cards/RFIDs to WSN nodes and proposed countermeasures [26] may mitigate these threats.

SCB-T14: Denial of Service (DoS) attacks on contactless smart cards by jamming communications signals. There may be significant similarities and applicability to WSN nodes and their communications.



WSN Node Threats

WSN Node Threat Reference	High Level Threat Description	Threat Applicable to Smart Cards (state whether contact or contactless)	Counter-measure Applicable to Smart Cards (state whether contact or contactless)
WSNN-T1	Dos, CoS & DCoS	✓(P) SCB	✓(P) SCB
WSNN -T2	Routing Data	×(N)	×(N)
WSNN -T3	Sybil & Sizzle	✓(P) SCA/B	✓(P) SCA/B
WSNN -T4	Routing Data	×(N)	×(N)
WSNN -T5	Routing Data	×(N)	×(N)
WSNN -T6	Routing Data	×(N)	×(N)
WSNN -T7	Possible 'C' Weaknesses in nesC	✓(P) SCA/B	✓(P) SCA/B
WSNN -T8	IEEE 1149.1 JTAG standard interface	✓(T) SCA/B	✓(T) SCA/B

Threat Totals	Countermeasure Totals
✓(T) = 1	✓(T) = 1
✓(P) to (T) = 0	✓(P) to (T) = 0
✓(P) = 3	✓(P) = 3
×(N) = 4	×(N) = 4

Fig. 3. WSN Nodes Compared to Smart Cards Matrix

WSNN-T1: DoS which may have a partial applicability to contactless smart cards (e.g., Jamming). We also apply a new term of Cessation of Service (CoS). Because WSN nodes are battery powered, they are designed to exploit a sleep mode to conserve power. If the nodes undergo continuous operations they will drain the battery. A sustained DoS attack may lead to a final CoS attack in that the node uses up all of its power and is no longer enable to function. An attack spread over a Wireless Sensor Network is a Distributed Cessation of Service (DCoS) attack.

WSNN-T2: This involves routing data between nodes and hence as such has no current applicability to the typical use of smart cards today, this may change in the near future with Java Card 3.0 and grid computing concepts for smart cards [14, 15].

WSNN-T3: The Sybil attack seems specific to WSN nodes, however the issue of spoofing, masquerading or exploiting multiple identities is something that can be shared to a partial degree between WSN nodes and smart cards. Sun's SSSL (Sizzle) mini web server [27] for WSN nodes may secure communications enabling confidentiality and if used with TLS meet integrity requirements too.

WSNN-T4 through to WSNN-T6: This involves routing data between nodes and hence as such has no applicability to smart cards. See WSNN-T2 above.

WSNN-T7: This involves weaknesses in the underlying programming languages. nesC [28] which is a C derivative used to create Tiny OS (a leading operating system for WSN nodes). The applicability to smart cards is minimal but may relate to native functions that smart card manufacturers utilise within their cards before installation of widespread operating systems or platforms.

WSNN-T8: This threat and countermeasure maps directly onto SCA/B-T5. Many nodes examined by Becher, Benenson and Dornseif had a JTAG connector on the node board easily accessible [24]. Attackers with appropriate kit may take control of the WSN Node.

6 Conclusion

This paper proposed a framework and methodology for classifying and analysing threats against smart cards and WSN nodes. Indications are that many attacks against smart card integrated circuits apply to WSN nodes and some WSN node RF/Communications attacks may apply to contactless smart cards and RFIDs.

Tamper resistance features within smart cards should be considered for WSN nodes. We suggest the need to establish High, Medium and Low assurance benchmarks for WSN nodes offering differing levels of security relative to use. Many technologies have matured through schemes like Common Criteria [2]and the production of Protection Profiles may help focus the development of security within WSN nodes.

Threats within WSN nodes shared with smart cards (specifically contactless smart cards) lie within the Radio Frequency space. However, communication and routing attacks are more effective against WSN nodes compared to smart cards due to the 'networked' nature of these attacks.

This paper has also defined two new definitions for attacks, 'Cessation of Service (CoS)' and/or a 'Distributed Cessation of Service (DCoS)' which may have wider applicability than just WSN nodes.

Overall, we feel that this 'path-finder' research has established the need for thorough scientific testing to prove or disprove the assertions made in this paper.

Other research areas that may closely tie into this research are suggested below:

Investigate RF/Communications threats between WSN nodes and Mobile Cell Phones for similarities (e.g., Bluetooth [29], IEEE 802.15.4 [1] and also ZigBee [30]).

A study of WSN nodes and sensor technologies in airports to assist baggage and passenger screening.

An assessment of smart card services/functionalities such as Global Platform [31] and Card Manager [32], Java Card Runtime Environment (JCRE) [33] and smart card APIs to determine applicability to WSN nodes.

The proposed framework and methodology in this paper may help to assess any shared security issues between Java Card 3.0 [13], secure distributed computing on a Java Card grid [14] and also the use of Active-RFID [34] and Passive-RFID [35] (which have an onboard power supply) and WSN nodes.

Authentication of WSN nodes is an often quoted security challenge [36, 37, 38]. The exploration of Attribute Certificates [39] and/or Kerberos tickets may enable novel secure authentication methods.

We are interested in investigating the potential for a secure authentication and routing protocol similar to IPSEC which we have provided a working label of KAFKA (Know Allies & Family, Know Adversaries) to suit the adaptive nature of Wireless Sensor Networks.

References

1. Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4 (Second Edition), http://shop.ieee.org/ieeestore/Product.aspx?product_no=SP1150
2. Common Criteria website: http://www.commoncriteriaportal.org
3. Smart Card Security User Group Smart Card Protection Profile (SCSUG-SCPP) Version 3.0 9 (September 2001)
4. JTC 1/SC 17: ISO/IEC 7810:2003 Identification cards - Physical characteristics (December 2005) http://http://www.iso.org/iso/en/prods-services/ISOstore/store.html
5. JTC 1/SC 17: ISO/IEC 7813:2006 Information technology -Identification cards – Financial transaction cards (June 2006) http://http://www.iso.org/iso/en/prods-services/ISOstore/store.html
6. JTC 1/SC 17: ISO/IEC 7816-1 to 15 Identification cards – Integrated circuit(s) cards with contacts (Parts 1 to 15) http://http://www.iso.org/iso/en/prods-services/ISOstore/store.html
7. JTC 1/SC 17: ISO/IEC 14443-1 to 4 Identification cards – Contactless integrated circuit(s) cards – Proximity cards (Parts 1 to 4) http://http://www.iso.org/iso/en/prods-services/ISOstore/store.html
8. JTC 1/SC 17: ISO/IEC 15693-1to 3 Identification cards – Contactless integrated circuit(s) cards – Vicinity cards (Parts 1 to 3) http://http://www.iso.org/iso/en/prods-services/ISOstore/store.html
9. German Federal Office for Information Security (BSI): Security Aspects and Prospective Applications of RFID Systems (2004)
10. Anderson, R., Kuhn, M.: Low Cost Attacks on Tamper Resistant Devices (1997)
11. Abraham, D.G., Dolan, G.M., Double, G.P., Stevens, J.V.: Transaction Security System. IBM Systems Journal v 30 no 2 (1991) 206-229
12. Skorobogatov, S.P.: Semi-invasive attacks - A new approach to hardware security analysis. PhD Thesis UCAM-CL-TR-630 ISSN 1476-2986 (April 2005)
13. Kocher, P. C.: Timing attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other systems. Crytpography Research Inc. CRYPTO (1996)
14. Java Card Forum: Working status & Deliverables Presentation (2005) http://www.javacardforum.org/03_documents/00_documents/marketingpresentationgeneral.pdf
15. Chaumette, S., Grange, P., Karray, A., Sauveron, D.: Secure distributed computing on a Java Card Grid - 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 7th International Workshop on Java for Parallel and Distributed Computing (2005)
16. Hollar, S.: Cots Dust Masters Degree Thesis (1996) http://www-bsac.eecs.berkeley.edu/archive/users/hollar-seth/publications/cotsdust.pdf
17. Tubaishat, M., Madria, S.: Sensor Networks: An Overview. IEEE Potentials, 22 (2003), 20–23.

18. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. Elsevier Article (2003)
19. Shi, E., Perrig, A.: Designing Secure Sensor Networks. IEEE Publication (2001). http://www.cs.ucsb.edu/~suri/Spr05/generalB.pdf
20. Benenson, Z., Freiling, F. C.: On the Feasibility and Meaning of Security in Sensor Networks. 4th GI/ITG KuVS Fachgesprch "Drahtlose Sensornetze", (2005) http://user.it.uu.se/~zina/publications/security-taxonomy.pdf
21. Healy, W.: Standards and Test Methods for Sensor Networks and Alert Systems (2007) http://www2.bfrl.nist.gov/projects/projcontain.asp?cc=8634508431
22. Official Website of the US Govt. Sensornet Project http://www.sensornet.gov/
23. Swiderski, F., and Snyder, W.: Threat Modelling, Microsoft Press (2004) 259
24. Becher, A., Benenson, Z., Dornseif, M.: Tampering with Motes: Real World Physical Attacks on Wireless (2005)
25. Finke, T., Kelter,H.: Abhrmglichkeiten der Kommunikation zwischen Lesegert und Transponder am Beispiel eines ISO14443 - Systems. BSI, (2004) http://www.bsi.de/fachthem/rfid/Abh_RFID.pdf
26. Rieback, M. R., Crispo, B., Tanenbaum, A. S.: Is your cat infected with a computer virus. Vrije Universiteit Amsterdam. (2006)
27. SUN's Sizzle (SSSL) Webserver for small devices http://research.sun.com/spotlight/2004-12-20_vgupta.html
28. Brewer, E., Culler, D., Gay, D., Levis, P., von Behren, R., Welsh, M.: nesC: A Programming Language for Deeply Networked Systems - (2004) http://nescc.sourceforge.net/
29. Official Website for the Bluetooth Short Range Wireless Connectivity Standardb http://www.bluetooth.com/bluetooth/
30. ZigBee Alliance Official Website. http://www.zigbee.org/en/index.asp
31. Global Platform Official Website. http://www.globalplatform.org/
32. Bernabeu, G.: GlobalPlatform Mobey Forum 2005 Presentation: Page 15 Card Manager Responsibilities (2005) http://www.globalplatform.org/uploads/Mobey%20Forum_Oct2005.pdf
33. Java Card Technology Datasheet (incl. JCRE) http://java.sun.com/products/javacard/datasheet.html
34. DoD Radio Frequency Identification Update: Enterprise Data Collection Across the Supply Chain (June 14, 2006). http://www.dla.mil/j-6/AIT/Files/Conferences/AirForce_Supply_Chain_AIT_Forum/2006_06_13/Day2/Smith%20-%20OSD%20RFID.pdf
35. United States Department of Defense Suppliers' Passive RFID Information Guide ver.8.0 (2005) http://www.acq.osd.mil/log/rfid/DoD_Suppliers'_Passive_RFID_Information_Guide_v8.0.pdf
36. Shaikh, R. A., Lee, S., Song, Y. J., Zhung, Y.: Securing Distributed Wireless Sensor Networks: Issues and Guidelines (2006) 226-231
37. Chan, H., Perrig, A.: Security and Privacy in Sensor Networks. IEEE Publication Computer, vol. 36, no. 10, (2003) 103-105
38. Muftic, S., Chang, C.: Security in Wireless Sensor Networks: Status, Problems, Current Technologies and Trends. Enisa Quarterly No. 4 (2006) 5-6
39. Arnab, A., Hutchison, A.: Ticket Based Identity System for DRM (2006)

Mobile Phones as Secure Gateways for Message-Based Ubiquitous Communication

Walter Bamberger[1], Oliver Welter[1], and Stephan Spitz[2]

[1] Technische Universität München, Germany
[2] Giesecke & Devrient GmbH, Germany

Abstract. For ubiquitous communication self-organising adhoc networks become more and more important. We consider mobile phones as an appropriate trusted gateway for external machines with low communication needs. A message-based approach is best in such a scenario with moving mobile phones and machines. We propose a security model for access control to the communication infrastructure that is also message-based. To meet the requirements of ubiquitous communicating machines, all algorithms on the sender's side are based on symmetric cryptography resulting in low computation needs. A sophisticated symmetric key infrastructure for message authentication provides the necessary key management. The trustworthiness of the mobile phone is achieved by using the SIM as a secure storage and computing module. This makes it possible to use the mobile phone not only as a user terminal but also as a trusted infrastructure component of the mobile network.

Keywords: SIM, mobile network, machine-to-machine communication, symmetric key infrastructure, message-based communication.

1 Introduction

2G/3G mobile networks with packet transport capabilities are widely spread today. They are also used for machine-to-machine communication. This paper introduces a security architecture for a communication technology, in which the external (sending) machine is equipped with a personal area radio (PAN, like ZigBee of Bluetooth) instead of a wide area radio (WAN, like GRPS or UMTS). This keeps the module complexity on the sender's side as well as the resource allocation in the mobile network very low. Interesting applications include all sorts of vending machines, escalators or environmental sensors.

Figure 1 illustrates the communication architecture considered in this paper. An external machine (on the left hand side) wants to send a message to a host in the Internet (e.g. running a web service). For this it looks for a randomly passing mobile phone and uses it as a relay. We call such a mobile phone the *gateway* in the following. In the mobile network there is another intermediate component named *proxy*. It performs accounting and security actions. In this paper we only discuss the unidirectional case from the external machine to the Internet host, although a bidirectional extension can be imagined.

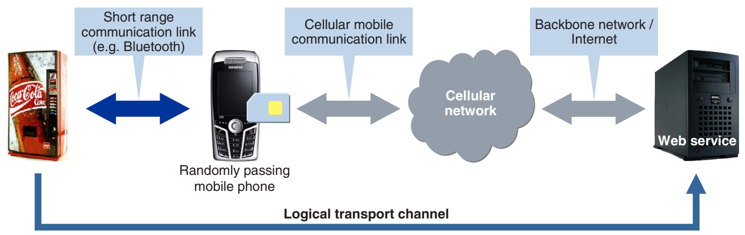

Short range communication link (e.g. Bluetooth)
Cellular mobile communication link
Backbone network / Internet
Cellular network
Web service
Randomly passing mobile phone
Logical transport channel

Fig. 1. The considered communication scenario: An external machine should be able to send messages supported by a trusted mobile phone

This paper deals with the security concerns that come along with this new approach. As the communication is message-oriented with one or more hops, a message-based security concept (Sect. 4.3) must be chosen. We show how this paradigm can be integrated in the existing security architecture of the mobile network. Message authentication is done using *symmetric keys* backed by a light-weight key management system (Sect. 4.2 and 5). A public key infrastructure like X.509 is not feasible as external machines have very low computation capacities and miss some prerequisites like access to a reliable time source. The proposed system respects the necessity of an easy deployment and allows implementing a simple application on present cheap hardware. For example the software could directly be implemented on the integrated micro-controller of the Bluetooth transceiver (like the BlueCore 4 of CSR). Then the hardware costs will be very low compared to other solutions.

The subscriber identity module (SIM) as a key component of the mobile network security serves as a key component in this new concept too. Because the gateway should operate as an external security wall preventing unauthorised traffic in the mobile network, the functionality of the gateway is split into a trusted and an untrusted part (Sect. 4.3). The SIM provides the trusted environment for storing the secret keys and does the security relevant calculations. The untrusted component handles the hardware access and is executed in the main processing unit of the mobile phone.

2 Related Work

There are interesting activities in the research community to enhance today's mobile networks with relaying techniques. The goals are mostly coverage extension and capacity improvements at moderate costs. Pabst et al. [1] provide a good starting point. We specialise our concept on machine-to-machine communication only.

For security related concepts a look at adhoc networks is also interesting. There are several proposals [2,3] to meet the adhoc nature with asymmetric cryptography and secret sharing techniques. Yang et al. [4] introduce a very localised and self-organising approach. However they do not really meet the characteristics of our communication system. Further more we want to evaluate the chances of symmetric cryptography.

Therefore a closer look at existing symmetric key infrastructures (SKI) can help for inspiration. The classical Needham-Schroeder protocol or Kerberos, but also newer proposals of Crispo et al. [5] target at user / machine authentication though. Some investigations show that this is rather different from a symmetric key infrastructure for message authentication with its keys which are shared by many devices.

Most closely related to our architecture, protocol and applications is the work of the Delay Tolerant Networking Research Group (DTNRG) in the Internet Research Task Force (IRTF). The main protocol is the Bundle Protocol [6], accompanied the Bundle Security Protocol [7]. Both are still drafts. The protocol is much more complex targeting more applications. However the security protocol still has a couple of open issues, especially the key management. With our simpler protocol we can provide a thorough and practical solution.

3 Technical Fundamentals

The mobile phone consists of three logical parts which are involved in the data exchange. The first hardware component is the insecure communication unit of the device responsible for the Bluetooth, NFC or IrDA communication with the external machine. The SIM module acts as a trusted storage and run-time environment for the security critical processes. The third component is the communication interface to the outside world, in our case the GSM network as a connection towards the Internet.

The SIM is the security critical gateway between both sides and is therefore responsible for all security related tasks. Data received from the insecure communication unit are verified, authorised and sent in a secure manner into the mobile network by the SIM. The communication to the SIM can be established via the classical APDU interface according to ISO 7816 or via a TCP/IP protocol stack on top of an USB connection to the SIM.

As we show in Sect. 5.1 the SIM must receive sensible key material from a server in the mobile network. Using the latest generation of Internet-enabled SIMs (like the Giesecke & Devrient GalaxSIM) a direct transport layer security (TLS) tunnel can be established between the server and the SIM. Then the mobile phone simply acts as a router between the SIM and the server. In case of an APDU based communication all data is routed through the insecure mobile phone operating system. Then additional security mechanisms have to be applied on the application level. We detail them in Sect. 5.1

The packet data protocol context (PDP context) [8] is another concept in 2G/3G networks that is important for this paper. A mobile phone, which wants to send packet switched data (e.g. via the general packet radio service (GPRS)), must request a packet data protocol context first. This context can be imagined as a virtual channel. A network protocol (e.g. IP), an interface address (e.g. an IP address) and other information is associated with this virtual channel. This also includes specific routing and charging rules. In our system the mobile phone requests a certain PDP context to deliver messages to the proxy in the mobile network. Using this PDP context the routing to the proxy is possible and the data transport is not charged to the mobile phone owner's account.

Because the PDP context is requested from an early component in the core network (the serving GPRS support node (SGSN)), refusing the PDP context for a given device is an efficient way to keep unwanted traffic to the proxy (which is free of charge) out of the mobile network.

4 Securing the Transport

The main purpose of this new communication approach is the message transport. Therefore this chapter discusses security aspects of the message transmission process. Starting with the required security services, the necessary symmetric key infrastructure is outlined next. With this background the message transmission process can be explained. The final section details a few important topics further.

4.1 Security Services

Talking about necessary security services corresponds with compiling the security requirements of the system. Therefore we first discuss these security services and show the realisation afterwards. For example Zhou and Haas [2] give the fundamentals.

In this scenario the data should not be transmitted through an end-to-end connection. Instead a message should be forwarded using one or more relays to reach its final destination. Each relay must verify the message *integrity* and whether it is allowed to use the infrastructure (*authentication*). This makes some kind of message authentication necessary.

Because the transmission in the mobile network causes costs, the mobile network operator must ensure the *non-repudiation* of origin. Another key infrastructure is set up for non-repudiation purposes, as the requirements are very different from the ones for message authentication. Note that, using symmetric keys, the mobile network operator can only prove that the message has not been created by a third party as he is able to create verifiable messages himself. A trust relation between the machine operator and the mobile network operator is assumed, so this will not become a problem.

Finally the *anonymity* of the mobile phone outside the mobile network must be ensured. We also increase the *availability* through redundancy: An external machine may re-transmit a packet several times depending on the booked service level. It should give attention to use different gateways for each re-transmission for security reasons.

Our system provides *confidentiality* too, but as an optional feature. There are a few applications that do not need this service but want avoid the extra effort.

4.2 Key Infrastructure

As mentioned in the previous section two sets of symmetric keys are used. With the *access control key set* each relay and the proxy can verify that a message (i.e. the sender) is authorised to use this mobile network for message-based communication. The *non-repudiation key set* is necessary for accounting purposes. With these keys the mobile network operator can determine the creator of the message uniquely. They are also used for packet encryption.

The access control key set consists of 32 keys. Each key is valid for a chosen time period (e.g. 3 years), and is replaced by a successor afterwards. It is identified with an identification number and a version number. Section 5 describes the key management for the access control key set further. The proxy in the mobile network has access to all 32 keys.

The access control key set is divided into two subsets of 16 keys each. A gateway has the keys of one subset, resulting in two types of gateways depending on the actual subset. This ensures that the system still runs, even if all 16 keys of one mobile phone are compromised. The keys are deployed onto the subscriber identity module card (SIM card) and cannot leave it. This guarantees the secrecy of them, as the subscriber identity module is considered to be a rather secure key storage. Section 4.3 details further how this module is used as a security kernel in this architecture.

An external machine has six keys, three out of each subset. During connection establishment with the gateway a subset is negotiated. Actually those six keys are not specific for a machine but for all machines of a machine operator (one company). External machines and their operating companies are considered to be the major risk for the secrecy of the keys.

In contrast the keys of the non-repudiation key set are not shared between the machine operators and the gateways. Each machine operator has its own unique key. The proxy in the mobile network uses these keys to verify the sender for accounting purposes. The keys are versioned as well, but the update process is not automatic. Instead the keys are exchanged during other service tasks on-site (e.g. every 5 years), so a sufficient long overlap between two consecutive key versions is required. Using only one key per machine operator reduces the size of the key database compared to individual SIMs in GSM modules.

4.3 Message Transmission Process

With this key infrastructure we can describe the transmission process of a message in the following.

External machine to gateway: First the external machine needs an *access control message key* derived from a key out of the access control key set and a *non-repudiation message key* derived from the machine's non-repudiation key. These message keys are recomputed for each message and help in combination with a nonce to hinder attacks based on a large collection of data or on messages with the same payload but different keys. Because there is no end-to-end connection the message key must be generated with a pseudo-random function and parameters only depending on header respectively packet information (see Sect. 4.4). The secret keys the message keys are derived from are called *master keys* in the following. Another parameter is the nonce which is generated by the mobile phone to prevent replay attacks. Therefore the message keys must be computed after the external machine has connected to the gateway.

With the non-repudiation message key the external machine encrypts the payload first. The encryption is indicated through a certain value in the content type header, as it is optional – meeting the needs of a few applications. Then two message authentication codes (MACs) must be computed, the *access control MAC* for the relaying and the *non-repudiation MAC* for accounting (see Fig. 2). To avoid the necessity to perform the hashing over the payload twice, a modification of the HMAC algorithm [9] is introduced in Sect. 4.4. With this the non-repudiation MAC is based on both message keys while the access control MAC is a common HMAC over the whole message, including the non-repudiation MAC. This makes it possible that every gateway can test the integrity of the message and verify that the message is authorised for this service. In addition the proxy can prove that the sender address indicates the right customer.

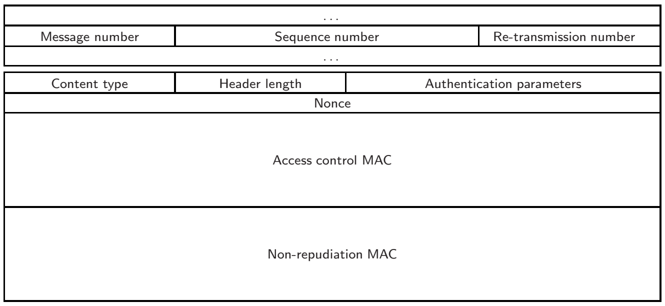


Fig. 2. Header of each packet

All in all when the external machine has found a gateway, it receives the number of the access control key set and a nonce, chooses an appropriate master key out of that key set, optionally encrypts the payload, computes both MACs and finally delivers the message to the mobile phone.

Processing within the gateway: In this concept the new extension to the security of the 3G network is the understanding of a mobile phone as a trusted gateway for message-based access. The trust originates from two measures: First we use the subscriber identity module as a secure key store and trusted processing platform, second each packet can be associated by the mobile network operator with a mobile phone and thus with a real world person.

Figure 3 shows that a server module in the main processing area of the mobile phone accepts the incoming messages from external machines. The symmetric keys for the access control MAC verification must be stored in a trusted environment. Therefore the server module forwards the message to the SIM card next. The method for the data exchange depends on the actual SIM card type. A small software module in the SIM verifies the access control MAC and sends it back to the main processor if the HMAC is valid. Otherwise it simply drops the message. This ensures that faked messages do not pass the mobile phone. The only chance for an attacker to send messages through the gateway consists in revealing a valid access control key. Compromise of the key will be detected at the proxy, because of a wrong non-repudiation key MAC. The key management system (see Sect. 5) provides methods for key revocation, so once the compromise is noticed, the abuse of the network is intercepted. All sensitive data is handled inside the trusted environment of the SIM and no secrets are visible from the untrusted domain at any time. This concept makes it obsolete to use an expensive trusted platform.

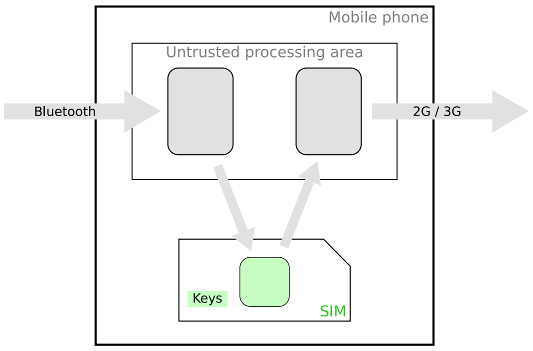


Fig. 3. Software architecture in the gateway

When the message passes the HMAC test in the SIM successfully, it is given back to another software module in the untrusted processing area of the gateway.

Gateway to proxy: To send the packet to the proxy in the mobile network, a software component in the untrusted area of the mobile phone first requests a specific packet data protocol context (PDP context) from the serving GPRS support node (SGSN). With this PDP context the mobile phone can access the proxy. It delivers the packet via an unsecured hypertext transfer protocol (HTTP) connection. Because a packet is usually much smaller than 100 kByte, the use of an authentication protocol like the transport layer protocol (TLS, [10]) would lead to a high overhead. It is more efficient to accept every packet, limit the packet size and verify both MACs. Therefore it is better in this situation to react effectively on attacks, instead of preventing them – although this channel into the mobile network is very vulnerable, because it is not charged to the mobile phone owner's account.

If one of the MACs or the combination of the non-repudiation key and the access control key is not valid (one non-repudiation key and exactly six access control keys are assigned to each machine operator), the proxy can detect an attack. The nonce and the various numbers in the header (see Fig. 2) make it possible to detect replay attacks. In addition optional destination filters at the proxy can protect companies if their non-repudiation key has been compromised. There are several measures available to react on those attacks:

- Traffic from manipulated mobile phones can be suppressed refusing the packet data protocol context for them. The mobile phone cannot send any data without charging anymore.
- Further criminal acts can lead to legal consequences, because the mobile phone owner is known.
- Attacks from devices behind the mobile phone are detected by the gateway. Only newly compromised keys could pass the gateway.
- Key management mechanisms as described in Sect. 5 make it possible to react precociously on compromised keys.

All in all we have seen that the use of the subscriber identity module as a trusted kernel combined with other existing security mechanisms of the mobile network makes it possible to keep unwanted traffic out of the mobile network. This architecture extends the 3G network efficiently for message-based external access.

4.4 Implementation Details

Message key computation: A limited number of 32 keys is used across many devices and many messages respectively packets. This makes it necessary to use a message key (k^m) for the MAC computation instead of one of those 32 master keys (k_i, $i = 1, \ldots, 32$) directly. The algorithm is the same for both, the access control message key and the non-repudiation message key. The only difference is the chosen master key.

The message key must be derived with a pseudo-random function (PRF) from a master key. In addition a nonce is necessary to randomise the message key. It is the nonce generated by the gateway and shown in Fig. 2. The function must be able to provide a bit stream with variable length depending on the actual hash function in the HMAC computation.

The pseudo-random function as defined in the draft of the Transport Layer Security protocol (TLS) v1.2 [10] is chosen here. The only difference is the absence of the label (see appendix A for convenience). The nonce concatenated with the source address, the destination address and the message number builds up the *seed*; the master key is used as the *secret*. A nonce may not be used twice, but it may be counted sequentially; the (pseudo-) randomness is provided by the PRF. All input values of the PRF are part of the header (see Fig. 2). Therefore all hops equipped with the master key can and should verify the access control MAC before forwarding the message.

Message authentication code computation: As Sect. 4.1 explains, two MACs are necessary for two different purposes: one to control the access to the relaying mechanism and another one to prove the origin of the message for accounting purposes. Using the conventional HMAC algorithm, this would result in two hash computations over the complete message. Since this system targets at external machines with low computation power, a modified combined method is proposed in the following. As a result the access control MAC can be verified with the usual HMAC verification algorithm, whereas the non-repudiation MAC needs both keys – the access control message key and the non-repudiation message key.

For the MAC generation an HMAC operation over the message m (without the not yet computed MACs) is performed with the access control message key k_{ac}^m first.

$$h_i = \mathrm{HMAC}(k_{ac}^m, m) \tag{1}$$

The non-repudiation MAC h_{nr} can be derived from this intermediate result with the non-repudiation message key k_{nr}^m:

$$h_{nr} = \mathrm{HMAC}(k_{nr}^m, h_i + nonce)$$

To verify this MAC both keys (k_{ac}^m and k_{nr}^m) must be known. This is true for the external machine and the proxy.

To complete the access control MAC, h_{nr} must be appended to the HMAC operation of (1). The state of that first HMAC computation must be preserved until this last HMAC computation. Then it is possible to verify the MAC with the usual HMAC algorithm over the complete message including the non-repudiation MAC, but in a slightly different order.

Both MACs can be inserted in the message as shown in Fig. 2.

5 Key Management

5.1 Access Control Keys

The system architecture as proposed in Sect. 4 relies on the secrecy of a set of keys for access control that is shared among all participants. In the following the proxy under control of the mobile network provider is considered equal with the central key management server.

Even if the main system uses symmetric cryptography, each subscriber identity module (SIM) contains an asymmetric key pair used for mutual authentication during key roll-out and key revocation.

Key roll-out: The SIM cards are delivered to the customers with an initial version of the secure application, an individual key pair and certificates necessary to authenticate themselves against the central server. On first start-up the subscriber identity module connects to the management system via a secure HTTP connection with mutual authentication. The asymmetric key pair is used for this. Through this secure tunnel it receives a current version of the software and the current key set.

The initial set of keys in the external machine comes with the hardware roll-out; thus the keys leave the protected environment of the network operator. This deployment is a very critical task but not in the scope of this work. Section 5.2 details further thoughts on this topic.

Key renewal: To allow key versioning each key index is extended by an additional version number. A new version number is the increment-by-one of its direct predecessor value. This enables the devices to decide if a presented key is newer or older than the one it currently uses without having access to the whole key history. In addition each key is associated with an expiration date. This time information is not security critical but is one way to trigger a key renewal procedure in the gateway.

The key renewal is done in two steps: In step I the new keys are made available on the central management node, from where the gateways can fetch them. A mobile phone starts the update procedure, when the expiration date has been exceeded or when a delivered message is rejected by the proxy because of an outdated key.

First the gateway sends a list of the key versions in its local key store to the server via an HTTP connection. The server compares the list with the key version in the repository and returns updates for all keys which posses a version difference of one. In this key renewal response the new key is encrypted with its predecessor, so no further authentication or transport encryption needs to be done (for details about the key renewal response see Sect. 5.3). The device must store the new key and the key renewal response for later use. If the distance between the key versions is larger than one or if the outdated key is comprised, the mobile phone must fetch the latest key and its key renewal response using the schema described for key revocation below.

In step II the new keys are distributed to the external machines. When a machine sends a message to the gateway, the key version of the message is examined and – if the version number in the local store is higher – the communication is instantly rejected. To distinguish such a rejection from other communication problems, a special (OBEX) error code is sent. The machine then requests a key update and receives a key renewal response as described in Sect. 5.3.

Again only a difference of one in the version number can be bridged by this mechanism. A larger gap would need an on-site service (compare Sect. 5.2). In the meanwhile the machine could use one of the remaining five keys. If the key version presented by the machine is newer than the one in the mobile device, the communication request is accepted but the message is kept in a quarantined state. As soon as a connection to the management system is available, a key update is performed and the message is evaluated using the new key.

Because the adhoc connection between the external machine and the gateway is very short-lived, some further considerations are necessary about the software architecture in the mobile phone. The access to the subscriber identity module is too slow. Section 5.3 details this further.

Key revocation: If one of the keys becomes compromised, any communication that is secured with that key must be rejected by the gateway. To signal the key invalidation to the participants in the network the version number is incremented by two to distinguish normal and revoking updates. The procedure for a revocation and a key renewal in case of a version difference larger than one is the same. The double increment forces the gateways to fetch the key update via an authenticated connection and breaks the update path for external machines.

In case of a revoked key, the subscriber identity module in the gateway establishes a secure HTTP connection with mutual authentication similar to the key roll-out procedure. Through this tunnel the SIM directly receives the new key and a key revocation note (see below). The latter one is forwarded to the untrusted area in the mobile phone to notify external machines about the key revocation.

There are a couple of ways to notify the gateways about compromised keys. The most promising methods are push messages like short message system (SMS) messages and notifications during message delivery. In the latter case the proxy informs the mobile phones about newly compromised keys every time they deliver a message (with this or another key). This can be done during the first three months for example. It seems to be a good heuristic as very active gateways are informed very fast this way without a traffic overhead.

There is no secure way to update compromised keys inside the external machine. The knowledge of the other keys is not sufficient to receive the new version of the key. Even if a key exchange is not possible, it is wise to push a key revocation note to the machine, so it no longer sends messages with an invalid key. This key revocation note is presented to machines using the compromised key, and it is secured by an HMAC with the compromised key. It is safe to use the compromised key to authenticate the key revocation note as an attacker may also send this note with all its keys he has got.

5.2 Non-repudiation Keys

The non-repudiation keys are known only to two parties – the machine operator and the network operator. Therefore a complex key infrastructure as introduced above is not necessary here. Instead these keys are considered to be more long-lived. If we assume that a service technician comes on-site at least once in two years, the key renewal process does not lead to an additional effort.

The key deployment demands a secure process within the company of the machine operator. It depends strongly on the organisational structure there and is therefore out of the scope of this work. Some thoughts on it include, that all keys (the non-repudiation and the access control keys) reside in an encrypted form on a cheap exchangeable flash memory (something like SD cards). All machines have a super key in their fixed flash to access their keys. This way it the keys do not leave a certain area in the company unencryptedly.

5.3 Implementation Details

Key renewal response: For each outdated key the server sends a new key encrypted to the gateway. The encryption is the bit-wise difference between the old and the new key: $u = k_i^n \oplus k_i^{n-1}$. To proof the authenticity of the update message, an HMAC using the old key is appended $r = u + \text{HMAC}(k_i^{n-1}, u)$. If the HMAC is valid, the update message is considered authentic and the gateway can recover the new key k_i^n with another XOR operation.

The above response is saved to be used for the key renewal between the gateway and the external machine as well.

Software architecture in the gateway for the key renewal: The key renewal between a gateway and an external machine imposes some problems based on the nature of adhoc networks. The time slot available for communication between the subscriber identity module and the machine can be very short and dispatching a message from the Bluetooth stack to the trusted execution environment on the SIM card has a high latency. To provide a fast response on key version errors and for key renewal responses, the version list and the encrypted key material is stored (in copy) in the untrusted area of the mobile device. Then the gateway can immediately respond on messages with outdated keys and on key renewal requests. Because no unsecured confidential data is involved in this process, the update process can be executed over any untrusted media to any kind of gateway or external machine.

6 Discussion of Selected Attacks

This section focuses on the vulnerabilities the above system imposes. Attacks on Bluetooth ([11,12,13] are good points of entry) or the mobile network are out of the scope because these technologies are present with or without our system and those attacks are mostly implementation dependent.

A major thread on today's communication systems are denial-of-service (DoS) attacks as they tend to be easy to execute. Looking at the external machine such an attack could be executed by simulating a legal gateway and capturing all packets a machine wants to send. Two methods could be combined to prevent this. First, it is part of the communication concept, that an external machine may send a message several times according to a booked service level. For retransmission different gateways should be used to complicate a successful attack. Second, machine operators who need a very high service level could configure their machines to authenticate the phone (with the access control keys). Because this costs much time, this decision should be well considered.

Next someone could try to attack the MACs of a captured message. To hinder this, message keys have been introduced. But basically it depends on the hash function, whether such an attack is possible. The HMAC specification [9] details the requirements for an appropriate hash function.

The next component is the gateway. All kinds of faked messages (including replayed messages) can be detected by the mobile phone, if it chooses the nonce appropriately. Only wrong non-repudiation MACs cannot be found. However the attacker must know access control keys in this case.

The application on the subscriber identity module must be written with security in mind, as a successful attack on it might reveal a whole subset of access control keys and possibly one authentication key pair. In general we consider a successful attack on the card hard but possible. However the keys on the SIM are not sufficient to successfully send a message into the Internet. A non-repudiation key is necessary too. Therefore the economic benefit in attacking a SIM is limited.

Finally the MACs, the combination of the keys, the nonce and the various numbers in the header of each message help to detect all kinds of attacks on the proxy in the mobile network. Revoking compromised keys and refusing the PDP context for the affected gateways are effective measures in this situation (compare with the end of Sect. 4.3).

Spreading a shared secret over many entities increases the probability of a compromisation. Alternatives like asymmetric cryptography or a significantly increased number of keys have many other downsides. Therefore we designed a dynamic system (with key renewal and revocation) which keeps nearly unaffected if either a SIM or an external machine is compromised.

7 Conclusion

This article proposed a security concept to extend the present 2G/3G network for message-based communication. It enables three interesting communication features: The asynchronous transfer provides a communication service even in areas without direct network coverage (the handset can carry the message into mobile coverage). The trust relationship between the external machine and the mobile phone is of a kind, that every user can become a potential node in this relay network. And finally the accounting and key infrastructure is company-based, leading to a minimal resource allocation in the mobile network.

To realise this we introduced a symmetric key infrastructure appropriate for message authentication in a relay network. It is supported by asymmetric keys which are only used for the initial enrolment and to renew compromised keys. All algorithms used in the external machines have been chosen to work with very low computation power.

The whole key management system was designed to work as a stand-alone solution, so a third party can provide this service using well defined interfaces to the mobile network operator. Nonetheless a good integration into the mobile network has been achieved, especially by using the SIM card as a secure storage pre-configured by the mobile network operator. The security level is similar to that of existing mobile networks.

If this service should be delivered by the mobile network operator only, the existing key infrastructure of the mobile network can take over some parts of the presented infrastructure. This will be subject of future work.

References

1. Pabst, R., Walke, B., Schultz, D., Herhold, P., Yanikomeroglu, H., Mukherjee, S., Viswanathan, H., Lott, M., Zirwas, W., Dohler, M., Aghvami, H., Falconer, D., Fettweis, G.: Relay-based deployment concepts for wireless and mobile broadband radio. Communications Magazine, IEEE **42**(9) (September 2004) 80–89
2. Zhou, L., Haas, Z.: Securing ad hoc networks. Network, IEEE **13**(6) (Nov/Dec 1999) 24–30

3. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing robust and ubiquitous security support for mobile ad-hoc networks. In: Ninth International Conference on Network Protocols, IEEE Computer Society (November 2001) 251–260
4. Yang, H., Meng, X., Lu, S.: Self-organized network-layer security in mobile ad hoc networks. In: WiSE '02: Proceedings of the 3rd ACM workshop on Wireless security, New York, NY, USA, ACM Press (2002) 11–20
5. Crispo, B., Popescu, B., Tanenbaum, A.: Symmetric key authentication services revisited. In: Information Security and Privacy. Volume 3108/2004 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2004) 248–261
6. Scott, K., Burleigh, S.: Bundle Protocol Specification (Internet draft). IRTF. (December 2006)
7. Symington, S., Farrell, S., Weiss, H.: Bundle Security Protocol Specification (Internet draft). IRTF. (October 2006)
8. 3rd Generation Partnership Project: 3GPP TS 23.060 V7.3.0: General Packet Radio Service (GPRS); Service description; Stage 2. (December 2006)
9. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104. Internet Engineering Task Force. (February 1997)
10. Dierks, T., Rescorla, E.: The TLS protocol. Version 1.2. Internet Engineering Task Force. (October 2006) draft.
11. Bluetooth SIG: Security. Web page (2007) http://www.bluetooth.com/Bluetooth/Learn/Security
12. Bialoglowy, M.: Bluetooth security review, part 1. Web page (April 2005) http://www.securityfocus.com/infocus/1830
13. Bialoglowy, M.: Bluetooth security review, part 2. Web page (May 2005) http://www.securityfocus.com/infocus/1836

A Pseudo-random Function for Message Key Generation

This paper uses the pseudo-random function (PRF) of the draft of the transport layer security (TLS) standard v1.2. It only omits the label. Then the function reads as follows:

$$\begin{aligned}\mathrm{PRF}(\mathit{secret}, \mathit{seed}) = {} & \mathrm{HMAC\_hash}(\mathit{secret},\ \mathrm{A}(1) + \mathit{seed}) + \\ & \mathrm{HMAC\_hash}(\mathit{secret},\ \mathrm{A}(2) + \mathit{seed}) + \\ & \mathrm{HMAC\_hash}(\mathit{secret},\ \mathrm{A}(3) + \mathit{seed}) + \ldots\end{aligned}$$

where hash must be substituted by a specific hash algorithm as defined in the chosen cipher suite (see authentication parameters field) and "+" is the concatenation operator. The function A is defined as

$$\begin{aligned}\mathrm{A}(0) &= \mathit{seed} \\ \mathrm{A}(i) &= \mathrm{HMAC\_hash}(\mathit{secret},\ \mathrm{A}(i-1)).\end{aligned}$$

An Information Flow Verifier for Small Embedded Systems[⋆]

Dorina Ghindici, Gilles Grimaud, and Isabelle Simplot-Ryl

L.I.F.L. CNRS UMR 8022
Université de Lille I, Cité Scientifique
F-59655 Villeneuve d'Ascq Cedex, France
{ghindici,grimaud,ryl}@lifl.fr

Abstract. Insecurity arising from illegal information flow represents a real threat in small computing environments allowing code sharing, dynamic class loading and overloading. We introduce a verifier able to certify at loading time Java applications already typed with signatures describing possible information flows. The verifier is implemented as a class loader and can be used on any Java Virtual Machine. The experimental results provided here support our approach and show that the verifier can be successfully embedded. As far as we know, this is the first information flow analysis adapted to open embedded systems.

Keywords: information flow, type checking, confidentiality, class loading.

1 Introduction

As the use of Java-enabled embedded systems such as smart cards, mobile phones and PDAs is growing, they are being associated with security since they provide a partial solution to the need for personal identification and non-repudiation. These devices evolve towards an open, multi-applicative environment supporting dynamic class loading and unloading.

Confidential data manipulated by such systems must be protected and accessible only by authorized users or programs. In a small open embedded system, where application may share code (e.g Api) or collaborate to offer better services, insecurity may stem from the code itself or from the code shared with some malicious untrusted software.

In order to enforce security, the Java Virtual Machine (Jvm) [17] and the Java Runtime Environment provide different mechanisms. For example, the bytecode verifier [16] uses static analysis to ensure that applications comply with the Java type system rules even for small systems [8,19], while the sandbox model is a dynamic mechanism, which enforces security by isolating applications. Java access modifiers (e.g. *public*, *private*, *protected*) express data accessibility for the Java language. Existing Java mechanisms control information access, and so

⋆ Funded by the Mosaïques project, CPER TAC 2005-2008 & FEDER.

they are not adequate in addressing data propagation. Ad-hoc mechanisms, like information flow verification, must be added to guarantee safe data propagation.

Despite the considerable amount of work on information flow achieved in the past decades, the information flow based enforcement mechanisms have not been widely used and applied in practice [26]. A survey on language-based security and information flow is presented in [20]. Most of the contributions in this area are based on static analysis [2,7,10,18,25] and on the type-based approach [3,13,22], where a type system is used to check secure information flow.

Low-level languages are taken into consideration only in few papers [3,15], while the Java language is rarely specified. In [4], only a small subset of the Java language is taken into consideration, while in [9], a compositional information flow analysis was implemented for mono-threaded Java bytecode. In [2], authors propose a static analysis similar to standard type verification used for Java bytecode. JavaCard features are considered in [5] and [11]. JFlow [18] is a powerfull tool, implemented as an extension of the Java language and structured as a source-to-source translator. It adds reliability to software implementation, but not to deployment and linking on a platform. The Pacap framework [5] involves a technique to verify interactions for Java enabled smart-cards, but the verification relies on the call graph, so it cannot be trusted in a Java/javacard open environment.

Unfortunately, the previous models focus on correctly checking information flow statically and do not address the challenge raised by an open computing environment.

In this paper, we propose an efficient model for detecting illegal information flows. Our model was successfully applied on small, open, Java-enabled systems. Our goal is to enforce data confidentiality for standard Java mobile code. In order to address the challenge raised by an open system, we enforce security properties at load time by performing a static analysis. Our implementation [23] works directly on Java bytecode and includes support for dynamic class loading and overloading.

The rest of the paper is structured as follows: Section 2 introduces some aspects of information flow and our approach. In Section 3 we present how we enforce confidentiality at load time on an embedded system. Section 4 describes how the information flow verifier can be integrated within a dedicated class loader in the KVM, but also how it can be used on any Jvm (e.g. JavaCard 3.0) as a user-defined class loader. Section 5 presents experimental results, while Section 6 summarizes our contributions.

2 Information Flow Analysis

2.1 General Aspects

Information flow stands for data propagation in a program. There are information flows arising from assignments (direct flows), from the control structure of a program (implicit flow), etc. For example, the code `p=s` generates a direct flow from `s` to `p`, while `if(s) then p=1 else p=0` generates an implicit flow. If `s` contains a secret, confidential value, and `p` a public, observable value, then the two examples are insecure and generate an illicit information flow, as confidential data can be induced by the reader of `p`. In literature, confidentiality is often seen as a non-interference [7,25] problem, as *public* outputs cannot depend on *secret* inputs. More exactly, for any initial value of an input *secret* variable, the values of *public* outputs do not change.

We target open (embedded) systems, allowing dynamic class loading, overloading and all the Java features, and supporting multi-applications sharing code. In this context, the insecurity for a class `A` may arise from the fact that `A` invokes an untrusted method `B.m` and it passes as argument a secret value. The system cannot guarantee that `B` does not make available the secret data. As our system must fit the Java dynamic class loading paradigm, we cannot use traditional approaches verifying the non-interference on the call-graph of a single application. In order to deal with openness, we perform a compositional analysis, computing for each method a stand-alone signature. The signature of a method is independent of the context under which the method is called. It contains the flows, potentially generated by the execution of the method, between elements that survive method execution: the method parameters, the method return value, an abstract value for the static world, one for exceptions, one for input/output channels. One "type" is associated with every element reflecting the flows generated by the method between this element and the others. Based on the knowledge of the flows, an application can check its own security policy. Thus, direct flows inside methods will be detected by traditional analysis while flows generated by interactions between methods will be detected by composition of the methods signatures. (Implicit flow inside methods is a natural extension of our approach, as the low complexity of the existing algorithms is promising for an embedded verification.)

2.2 Algorithm

We propose a model in which, as in classical information flow, each field is annotated by a security level, *secret* or *public*: a *secret* field should not be made accessible through information flow to unauthorized parties. Tracing all the fields of an object is expensive in time and memory and is not always possible when the calling context is not available. Moreover, imposing some kind of "subtyping" of signatures constrains the use of overloading. So, we split each object in only two parts, a secret part and a public part. The secret part of an object o, denoted by o^s, stands for all access paths starting from o which contain at least one field having the security level *secret*, while the public part, denoted by o^p is the complementary. Experimental results showed that our simplifying assumptions are reasonable in practice.

Considering our split of objects and the dichotomy of Java types (elementary types and object types), the links between two elements a and b have the form $a^{\wp(p,s)} \xrightarrow{v/r} b^{\wp(p,s)}$, where v denotes a value link, r a reference/alias link, s and p the secret or public part, while $\wp(p,s)$ denotes subsets of $\{p,s\}$. As a reference link includes the value link between the same elements, and as the *public* or *secret* part are included in the entire element, we can define an order relation between flows. Using this partial order relation, we obtain a lattice of links that contains 80 possible flows between two elements. The bottom of the lattice is represented by an empty set, meaning that there is no flow of information between a and b, while the top of the lattice is represented by $\{a^{p,s} \xrightarrow{r} b^{p,s}\}$, meaning that there is a reference link (alias) between the secret and public part of a and the secret and public part of b. More details on the type system and the lattice of links can be found in Appendix A.

Let's consider a class C having a *secret* field `s`, a *public* field `p` and a method `m`. Fig. 1 presents the signature of `m` at each point of the program, considering that the external method `foo` contains a value link from the return of the method, denoted by R, to the first parameter of the method ($R \xrightarrow{v} p_1$).

	`void m(int x, A a) {`		
1		`iload_1`	
2	`if(x>0)`	`ifle 6`	
3		`aload_0`	
4		`iload_1`	
5	`this.s = x;`	`putfield C.s`	$S_m = \{this^s \xrightarrow{v} x\}$
6		`aload_0`	
7		`aload_2`	
8		`iload_1`	
9		`invokevirtual A.foo`	
10	`this.p = a.foo(x);`	`putfield C.p`	$S_m = \{this^s \xrightarrow{v} x, this^p \xrightarrow{v} x\}$
11	`}`	`return`	with $S^e_{foo} = \{R \xrightarrow{v} p_1\}$

Fig. 1. Example

In order to ensures threads-safety, the abstraction for static elements has the default security level *public*, as all the *secret* data linked to static attributes and susceptible to be accessed through different threads are considered leaky

To compute methods signatures, we perform for each method an intra-method static abstract interpretation relying on a classical operational semantics composed of a set of transformation rules. The abstract values are represented by elements composing the signature of the method, and some other internal values needed to correctly analyze the method. The analysis does not rely on the call graph: the interpretation of an *invoke* bytecode consists of applying the signature of the called method to the signature of the calling method.

Each instruction has associated an abstract state representing the state before executing the instruction. The state contains the local operand stack, the local variables, and the current signature of the method at an execution point. This state contains the union of all possible states under which the associated bytecode can be executed. The control flow structure of the Java bytecode dictates an iteration on the set of instructions for each method. At each iteration, the current bytecode is abstractly interpreted and the resulting state is merged with the state already associated with its successors. For *invokevirtual* bytecodes, when the exact type of the called object cannot be statically determined, we take into consideration a global signature which is the union of all possible signatures for the desired method implemented in the class hierarchy derived from the static type of the object. Since the number of abstract values is finite and we perform merge operations, a fixed point is reached.

The existence of recursive and inter-depended methods dictates an incremental inter-method analysis, starting with the set of empty signatures and iterating on a set of classes until a fixed-point has been reached. This allows us to obtain more precise results.

Noninterference is too restrictive for common applications such as cryptographic functions, where outputs often depend on secret outputs. However, one should not be able to derive the secret from the output. To handle this and other intentional release of secret data, the proposed systems allows to manually annotate trusted methods with the desired signature. A more precise approach would be to include a mechanism for declassification [21,12] in order to specify what information can be released and where.

3 Information Flow Verifier

In the previous section, we presented a compositional information flow analysis complying with the Java dynamic class loading paradigm. But experimental results, as depicted in Fig. 4, show that at least 3 iterations must be performed on the set classes and an average of 1.5 iterations on the set of bytecodes for each method. Therefore, the analysis is already expensive for a normal system and impracticable for a device having limited resources.

In the context of small objects, a technique known as "Lightweight bytecode verification" (LBV) [19] has been developed for Java bytecode type verification. This technique, closely related to proof-carrying code [6], lies on the simple idea that it is easier to verify a result already computed. A small device can verify code received from an untrusted source without relying on a third party even if it has not enough power to compute the proof itself. It is based on two phases: an external phase which computes the type correctness and annotates the bytecode with some proof elements, and an embedded phase, which verifies, at loading time, the annotations obtained during the external phase. The verification operation is linear in code size and uses constant memory. The off-board analysis and the proof can be computed by any device or tool, as the small device can verify the code it receives without relying on the external device. LBV relies on the lattice structure of types and on unification operations on this lattice. The lattice of links allows us to use this technique in our context. While LBV checks explicit Java types, our algorithm has to infer information flow links. We have to deal with type inference and with signature management.

3.1 Signature Computation

The verification process is performed while loading a Java class. In order to ease verification, we ship with each class C some proof: the state of the JVM for each target instruction in each method, the signatures of methods invoked in C, the security levels of fields used in C. The proof elements are defined as new attributes of the `class` file structure, so the annotated classes can be loaded by any JVM, even by those not enforcing information flow security properties.

Due to limited resources of embedded systems, the size of proof elements must be as small as possible. As the lattice of links contains 80 possible flows, we chose a binary and compact solution on 1 byte to encode the links. This solution allows simple manipulation operations. For example, adding a new link to a signature requires only a binary logical operation. Moreover, the signatures within the states of the JVM for each target instruction are encoded incrementally: the first signature is encoded, while the subsequent signature is defined by the the flows added or deleted in the previous signature. Experimental results showed that signatures have few changes from one label to another.

The verification consists in a sequential interpretation of bytecodes of each method of the class. When a target bytecode is found, the current state of the JVM must be compatible with the proof corresponding to the target bytecode: if the compatibility is not tested, the class is rejected; otherwise the verification is carried on using the proof as the current state of the JVM. Given two states A and B of the verification process, A is compatible with B if the stack and local variables are compatible (state B contains at least all the elements in A) and the two signatures, S_A and S_B, are compatible. The stack comparison is possible, as we assume that the bytecode was already checked by the JVM verifier and thus it is well typed. A signature S_A is compatible with S_B if S_A contains at least all the links present in S_B, according to the lattice of method signatures, which is a natural extension of the lattice of links defined previously.

Dead code is ignored by the external analysis and thus not annotated. In order to deal with this situation, when a label bytecode without any proof annotation is found, we can assume it is the beginning of a block which is never executed. In this case, all the bytecodes following the label are ignored, until we meet a label with a proof. If the label without a proof is not the start of a dead block, then the class is rejected when the compatibility of predecessors instructions with the proof of the label is tested.

The embedded verification has the advantage that each instruction is interpreted only once and so it is linear in time with the code size. Moreover, the proof is used only during the verification and not stored in the system. Only the final signature of each method is kept on board. Another advantage is that each class is verified only once, even the code shared by many applications, as the signatures are kept on board in a dictionary. If the type inference of method signature fails, the class is rejected. If the type inference succeeds, we must ensure that the signatures used during validation fit within the system.

The analysis guarantees noninterference for loaded classes. All the possible flows from secret to public data are detected and present in the signatures.

But, due to our simplifying assumptions, we might detect false flows. Practical experiments showed that this situation does not occur very often.

3.2 Signature Management

Classes are loaded one by one. Once a class is loaded, the validated signatures are kept in a dictionary. In order to validate a class C at loading time, we load with C the signatures of all methods invoked in C (called "external methods").

When we load the class C from the example in Fig. 1, we will also load the signature S^{e}_{foo} of the method `foo` in the class A. If the class A has already been loaded, the external signature S^{e}_{foo} will be ignored and the signature of foo from the dictionary will be used while verifying C. If the class A has not been yet loaded, S^{e}_{foo} will be used while analysing C. If C is accepted, the signature S^{e}_{foo} will be kept on board into a temporary dictionary until the class A is loaded.

Let's assume that A is loaded later and the method foo has the signature S_{foo}. A will be accepted only if the signature S_{foo} is compatible with S^{e}_{foo}. The external signature S^{e}_{foo} should contain at least all the links from the loaded signature S_{foo}, otherwise the previous verification of C is not correct. If the external signature contains fewer links than the loaded one, it is acceptable, as long as we do not miss any information leakage. If the class A is certified and loaded on the system, S^{e}_{foo} and all the external signatures for A previously loaded are erased from the temporary dictionary.

There are different possible scenarios. We now consider the following:

load class C
  external method A.foo with $S'_{foo} = \{R \xrightarrow{v} p_1\}$
  store S'_{foo} in the temporary dictionary
load class D
  external method A.foo with $S''_{foo} = \{R \xrightarrow{v} p_1, this^s \xrightarrow{v} p_1\}$
  store infimum(S'_{foo},S''_{foo})=$\{R \xrightarrow{v} p_1\}$ in the temporary dictionary
load class A
  A.foo with signature S_{foo}

We load two classes C and D, and each one claims a different external signature for $A.foo$. As to validate the class C we use S'_{foo}, and to validate D we use S''_{foo}, the real signature S_{foo} should be compatible with S'_{foo} and S''_{foo}. All the flows in S_{foo} should be in S'_{foo} and S''_{foo}, which means that S_{foo} should be compatible with infimum of S'_{foo} and S''_{foo} according to the lattice of method signatures. So when we have different external signatures for the same method, we keep the infimum in the temporary dictionary.

The correctness of a signature depends also on the security levels of used fields. To have access to security levels of external fields of a class, we use a procedure similar to the one used to load the external methods. Two fields are compatible if they have the same security level. We also check the compatibility of loaded signatures with the global signatures belonging to extended classes.

4 The Verifier as a User-Defined Class Loader

The loading process in a JVM is performed by the class loaders. The standard JVM deals with multiple class loaders, hierarchically organized, and supports user-defined class loaders. The KVM virtual machine [24] does not support user-defined class loaders and has a single built-in class loader that cannot be overridden or replaced by the user.

We built a verifier that can be run on any JVM. It can be built in the single class loader of KVM or installed as a user-defined class loader for a standard JVM. The embedded JVM [1,14,24] are evolving towards the standard Java language, and therefore towards a multiple class loader hierarchy. The recently presented JAVACARD 3.0 does the same. We describe now how the verifier can be used as a plug-in within a standard JVM to validate annotated bytecode.

Scl_2 loads class C
    external method Scl_2.A.foo with signature S^2_{foo}
Scl_1 loads class D
    external method Scl_1.A.foo with signature S^1_{foo}
Scl_1 loads class A
    Scl_1.A.foo with signature S_{foo}

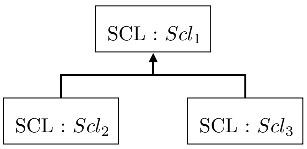


Fig. 2. Loading example in a *SafeClassLoader* hierarchy

Applications implement subclasses of ClassLoader in order to extend the manner in which the JVM dynamically loads classes. Class loaders may typically be used to check security properties. The verifier was implemented as a subclass of the *ClassLoader* class provided by the Java API, named *SafeClassLoader*. Certifying the underlying information flow of an application requires the instantiation of a *SafeClassLoader* with which the application should be loaded.

In the JVM delegation model, class loaders are arranged hierarchically in a tree, with the bootstrap class loader as the root of the tree. Each user-defined class loader has a "parent" class loader. When a load request is made by a user-defined class loader, that class loader usually first delegates the parent class loader, and only attempts to load the class itself if the delegate fails to do so. A loaded class in a JVM is identified by its fully qualified name and its defining class loader. This is sometimes referred to as the runtime identity of the class. Consequently, each class loader in the JVM can be said to define its own name space. In the same manner, each *SafeClassLoader* defines its own dictionary containing the signatures of loaded methods.

Let's consider a hierarchy containing three *SafeClassLoader*s, Scl_2, Scl_3 and their parent Scl_1, and the scenario in Fig. 2. Class loader Scl_2 requests to load class C. At first, it delegates its parent class loader, Scl_1, to load C. If the delegation fails, Scl_2 attempts to load the class by itself. While loading C, Scl_2 tries to find the signature of $A.foo$: it first searches in its dictionary, and if the search fails, it delegates the search to its parent, which repeats the procedure. If the parent also fails to find the signature, external signature S^2_{foo} is used while validating C and stored in the temporary dictionary. Class loader Scl_1 loads a class D also containing an external signature for $A.foo$. The external signature S^1_{foo} is stored in the temporary dictionary and is associated with Scl_1.

Finally, class loader Scl_1 attempts to load class A. Let S_{foo} be the verified signature of foo. Class A will be loaded by Scl_1 if and only if S_{foo} is compatible with the external signatures for foo in the current class loader (Scl_1) and with the external signatures in class loaders that can delegate Scl_1. In our case, S_{foo} must be compatible with S^1_{foo} and S^2_{foo}. Otherwise, class A is rejected.

In order to verify this kind of compatibility, external signatures must be accessible to all class loaders. This is why we implemented an unique temporary dictionary which is used by all class loaders. The example showed how the $SafeClassLoader$ extends the delegate model to the look up of a signature of a method. The same search process is extended to the look up of the security level of a field.

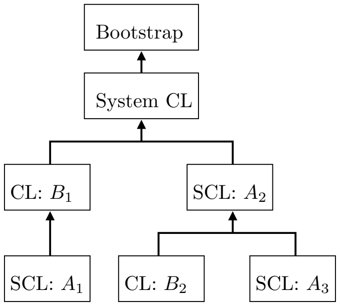


Fig. 3. Class Loader hierarchy example

We presented so far the case where all the class loaders in the hierarchy have the type $SafeClassLoader$. Actually, the hierarchy contains different types of class loaders. As shown in Fig. 3, the bootstrap class loader loads the classes from the JVM, as well as extensions to the JDK. The system class loader loads all the classes provided by the classpath. In the end, we have several additional class loaders, where SCL defines a $SafeClassLoader$ and CL any other type of class loader.

As a consequence, we must take into consideration the validation of classes loaded by any class loader. Let's consider that A_1 loads a class C that invokes a method of another class D already loaded by the parent B_1. As B_1 is not a $SafeClassLoader$, the classes it has loaded have not been validated at loading time. To ensure security for C, $SafeClassLoader$ A_1 will try to retrieve, using the *getResourceAsStream* method, the .class files of the classes loaded by its parent and to verify the announced signatures. If the streams cannot be found, or they do not contain information flow attributes, or the signatures are not compatible with the announced ones, A_1 rejects class C. Otherwise, the signatures of classes belonging to a non-$SafeClassLoader$ are stored in a special dictionary, named "system dictionary". The look up for a signature in a class loader is performed in its dictionary, if the class loader is a $SafeClassLoader$, and otherwise in the system dictionary.

In order to support any JVM, we do not interfere while the Bootstrap and System class loaders load the JVM and classpath classes, and thus we consider their signatures as part of our trust computing base.

5 Experimental Results

This section describes the results of experiments run on some significant benchmarks such as Dhrystone, a well known benchmark for embedded systems, The Fast Fourier Transform (FFT), a common signal processing application, crypt (a data encryption algorithm) and Pacap [5], an electronic purse case study for information flow checking (for which we detected the same illicit flows as in literature). We ran the experiments using a Java Runtime Environment, Standard Edition (build 1.5.0_09), on a Linux system running on a Intel(R) Pentium(R) M processor 2.13GHz with 1Gb memory.

			Off board analysis					On board verification				
Benchmark	Classes	Methods	Class iterations	Bytecode iterations	Analysis time (s)	Average memory (Kb)	Maximum memory (Kb)	Execution time CL (ms)	Execution time SCL (ms)	Verification time SCL (ms)	Average memory (Kb)	Maximum memory (Kb)
Dhrystone	5	21	3	1.35	3.1	2.66	35.80	111	373	310	0.36	2.05
fft	2	20	3	1.55	3.2	1.50	7.86	63	175	161	0.39	1.80
_201_ compress	12	43	3	1.84	4.4	2.02	20.68	321	522	251	0.37	2.31
_200_check	17	109	4	1.18	8.7	3.87	34.45	129	883	794	0.58	3.51
crypt	2	18	3	1.32	4.1	2.91	20.58	46	268	222	0.56	3.68
lufact	2	20	3	1.75	3.4	3.90	23.22	537	876	303	0.45	1.25
raytracer	12	72	5	1.53	4.6	1.30	25.10	80	440	422	0.39	1.80
Pacap	15	92	4	1.05	4.7	3.00	92.68	30	281	275	0.38	3.19

Fig. 4. Off board analysis and on board verification measurements

Benchmark	Initial class size (Kb)	Annotated class (Kb)	Signatures (%)	Labels proof (%)	External methods (%)	External fields (%)
Dhrystone	8.2	11.9	3.17	23.07	2.42	0.20
fft	6.8	11.3	4.53	48.47	5.17	0
_201_ compress	20.1	28.3	3.79	23.36	4.21	0.33
_200_check	46.3	80.3	4.62	57.25	3.82	0.05
crypt	7.0	12.3	6.04	58.34	4.66	0.19
lufact	9.3	14.3	3.37	43.00	2.06	0.40
raytracer	24.0	33.4	1.41	16.62	5.51	0.63
Pacap	26.8	36.9	5.71	18.37	3.46	0.38

Fig. 5. Size of annotations

First, we ran the external application computing the information flow signatures and annotating the classes (Fig. 4, Off board analysis) in order to find out how the algorithm performs in practice. We measured the number of iterations for the inter-method analysis (iterations on a set of classes), the iterations for the intra-method analysis (iteration on each meathod's instructions set) and the time needed to perform the analysis. The results showed that the computation algorithm is quite expensive in terms of time complexity: in average, we need 3 iterations on the set of classes, 1.5 iterations on the instruction set and 4.5s for each application. For the JVM *spec* benchmarks, we performed the library analysis before carrying out the experiments.

Secondly, we loaded the annotated applications generated by the off board analysis (Fig. 4, On board analysis). In order to find out how the JVM loading process is hampered by our verification, we measured the execution time in two cases: with (SCL) and without (CL) information flow verification. We observed that the verification implies an average execution time 3 times as large as the standard one. But the information flow verification is performed only once, at loading time, so any subsequent running of the applications is not hindered. Moreover, the average verification time (342.25ms) is more than 10 times smaller than the average analysis time (4.25s). As expected, the verifier performs much faster than the computation algorithm.

Lastly, we measured the size of the proof and the signatures loaded with the code (Fig. 5). The proof, the external methods and external fields represent 39.73% of the total size of initial .class files. This data is used only during the verification process, at loading time, and it is not stored on the device, so its size does not have a significant impact on the embedded system. The signatures, which are stored in the dictionary and kept in the system, make up only 4.01% of the initial .class size, an acceptable overhead.

6 Conclusion

Confidentiality represents a real concern in embedded systems manipulating sensitive data. Practical information flow models are almost non-existent, despite the quality of the underlying theory. As the algorithm for detecting illegal information flows in an application has a high complexity, we propose a lightweight verification for embedded systems. The information flow is certified at loading time, using some proof elements previously computed and shipped with the code. The information flow verification is performed in linear time and uses almost constant memory. Experimental results conducted for the external analysis and for the embedded verification support our approach. The time penalty and the memory consumption introduced by the verifier are acceptable. We think that we can cut by at least 50 percents the size of the elements embedded within the code by modifying the encoding of proof and signatures.

References

1. AONIX INC. Perc products.
2. AVVENUTI, M., BERNARDESCHI, C., AND FRANCESCO, N. D. Java bytecode verification for secure information flow. *SIGPLAN Not. 38*, 12 (2003), 20–27.
3. BARTHE, G., BASU, A., AND REZK, T. Security types preserving compilation: (extended abstract). In *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004* (2004), vol. 2937 of *Lecture Notes in Computer Science*, Springer, pp. 2–15.

4. Barthe, G., D'Argenio, P., and Rezk, T. Secure Information Flow by Self-Composition. In *Computer Security Fundation Workshop (CSFW'17)* (2004), IEEE Press, pp. 100–114.
5. Bieber, P., Cazin, J., Girard, P., Lanet, J.-L., Wiels, V., and Zanon, G. Checking secure interactions of smart card applets: extended version. *J. Comput. Secur. 10*, 4 (2002), 369–398.
6. Colby, C., Lee, P., Necula, G. C., Blau, F., Plesko, M., and Cline, K. A certifying compiler for java. In *PLDI '00: Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation* (New York, NY, USA, 2000), ACM Press, pp. 95–107.
7. Denning, D. E., and Denning, P. J. Certification of programs for secure information flow. *Commun. ACM 20*, 7 (1977), 504–513.
8. Deville, D., and Grimaud, G. Building an 'impossible" verifier on a Java card. In *Proc. 2nd USENIX Workshop on Industrial Experiences with Systems Software (WIESS'02)* (Boston, USA, 2002).
9. Genaim, S., and Spoto, F. Information Flow Analysis for Java Bytecode. In *Proc. of the Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)* (Paris, France, January 2005), R. Cousot, Ed., vol. 3385 of *LNCS*, Springer-Verlag, pp. 346–362.
10. Ghindici, D., Grimaud, G., and Simplot-Ryl, I. Embedding verifiable information flow analysis. In *Proc. Annual Conference on Privacy, Security and Trust* (Toronto, Canada, 2006), pp. 343–352.
11. Hansen, R. R., and Probst, C. W. Non-interference and erasure policies for java card bytecode. In *6th International Workshop on Issues in the Theory of Security (WITS '06)* (2006).
12. Hicks, B., King, D., and McDaniel, P. Declassification with cryptographic functions in a security-typed language. Tech. Rep. NAS-TR-0004-2005, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, May 2005.
13. Hunt, S., and Sands, D. On flow-sensitive security types. In *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006* (2006), ACM, pp. 79–90.
14. Java In The Small. http://www.lifl.fr/POPS/JITS/.
15. Kobayashi, N., and Shirane, K. Type-based information flow analysis for low-level languages. *Computer Software 20(2)* (2003), 2–21.
16. Leroy, X. Java bytecode verification: Algorithms and formalizations. *J. Autom. Reason. 30*, 3-4 (2003), 235–269.
17. Lindholm, T., and Yellin, F. *Java Virtual Machine Specification.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
18. Myers, A. C. Jflow: practical mostly-static information flow control. In *POPL '99: Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (1999), ACM Press, pp. 228–241.
19. Rose, E., and Rose, K. H. Lightweight bytecode verification. In *Workshop "Formal Underpinnings of the Java Paradigm", OOPSLA'98* (1998).
20. Sabelfeld, A., and Myers, A. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications, 21(1), 2003. 21*, 1 (january 2003).
21. Sabelfeld, A., and Sands, D. Dimensions and principles of declassification. In *CSFW '05: Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW'05)* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 255–269.

22. Smith, G., and Volpano, D. Secure information flow in a multi-threaded imperative language. In *POPL '98: Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (1998), pp. 355–364.
23. STAN - STatic Alias aNalyser. http://www.lifl.fr/~ghindici/STAN/.
24. Sun Microsystem. Connected Limited Device Configuration and K Virtual Machine, `http://java.sun.com/products/cldc/`.
25. Volpano, D., Irvine, C., and Smith, G. A sound type system for secure flow analysis. *J. Comput. Secur. 4*, 2-3 (1996), 167–187.
26. Zdancewic, S. Challenges for information-flow security. PLID'04 The First International Workshop on Programming Language Interference and Dependence, August 25 2004, Verona, Italy, August 2004.

A The Type System

For each element a from the set of abstract values we define the flow relation as a tuple composed of four elements: the security level of a (in $\wp(p, s)$), the type of flow (v or r for *value* or *reference*), the element to which a points to and its security level. The type associated with a is the union of all possible flows from a to b. For example, a *value* link from the *public*part of a to the *secret* part of b corresponds to the type (p, v, b, s). For convenience, we will denote this flow by $a^p \xrightarrow{v} b^s$.

If both public and secret parts of an element can be accessed, the whole element can be accessed. Thus, a link to the entire element ($b^{p,s}$) is greater than the same link to only one part of the element(b^s, b^p). Moreover, having access to a reference means having access to all its values. Thus, the *value* link is included in the *reference* link. Using this partial order relations, we obtain a lattice of links (Figure 6) having union as upper bound computation and inclusion as order relation.

$$\begin{tikzcd}
 & & a^{p,s} \xrightarrow{r} b^{p,s} \arrow[dl, dash] \arrow[dr, dash] & & \\
 & a^{p,s} \xrightarrow{r} b^s \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & a^p \xrightarrow{r} b^{p,s} \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
a^s \xrightarrow{r} b^s \arrow[d, dash] & a^{p,s} \xrightarrow{v} b^s \arrow[dl, dash] \arrow[dr, dash] & a^p \xrightarrow{r} b^s \arrow[d, dash] & a^p \xrightarrow{v} b^{p,s} \arrow[dl, dash] \arrow[dr, dash] & a^p \xrightarrow{r} b^p \arrow[d, dash] \\
a^s \xrightarrow{v} b^s & & a^p \xrightarrow{v} b^s & & a^p \xrightarrow{v} b^p
\end{tikzcd}$$

Fig. 6. Extract of the lattice of links

Survey and Benchmark of Stream Ciphers for Wireless Sensor Networks

Nicolas Fournel[1], Marine Minier[2], and Stéphane Ubéda[2]

[1] LIP, ENS Lyon - Compsys (Bureau 347)
46, Alle d'Italie - 69364 LYON Cedex 07 - France
`Nicolas.Fournel@ens-lyon.fr`
[2] CITI - INSA de Lyon - ARES INRIA Project
Bâtiment Léonard de Vinci
21 Avenue Jean Capelle, 69621 Villeurbanne Cedex - France
`FirstName.Name@insa-lyon.fr`

Abstract. For security applications in wireless sensor networks (WSNs), choosing best algorithms in terms of energy-efficiency and of small-storage requirements is a real challenge because the sensor networks must be autonomous. In [22], the authors have benchmarked on a dedicated platform some block-ciphers using several modes of operations and have deduced the best block cipher to use in the context of WSNs.

This article proposes to study on a dedicated platform of sensors some stream ciphers. First, we sum-up the security provided by the chosen stream ciphers (especially the ones dedicated to software uses recently proposed in the European Project Ecrypt, workpackage eStream [27]) and presents some implementation tests performed on the platform [16].

Keywords: stream ciphers, sensors, benchmarks.

Introduction

Sensor networks are made by the tremendous advances and convergence of micro-electro-mechanical systems (MEMS), wireless communication technologies and digital electronics. Sensor networks are composed of a large number of tiny devices or sensors which monitor their surrounding area to measure environmental information, to detect movements, vibrations, etc. Wireless sensor networks can be really useful in many civil and military areas for collecting, processing and monitoring environmental data. A sensor node contains an integrated sensor, a microprocessor, some memories, a transmitter and an energy battery. Sensor nodes communicate through a radio device in order to manage the network and to gather the produced data to a specific node called the sink node. Despite the relative simplicity of its basic components, sensor networking offers a great diversity: various hardwares (MicaZ, Telos, SkyMote, AVR or TI micro-controllers), various radio and physical layers (868MHz and 2,4GHz) using different types of modulations, various OS (TinyOS, Contiki, FreeRTOS, JITS), various constraints (real-time, energy, memory or processing), various applications (military or civil uses).

In such a context, a specific care must be invested in the design of the applications, communication protocols, operating systems and of course security protocols that will be used. Lots of protocols have been proposed to enforce the security offered by sensor networks. In despite of the increasing request in this new area of research, few articles presented real results of implementations or benchmarks concerning the security primitives which can be used in sensor networks. In [22], the authors present such results concerning theoretical aspects and benchmarks for the most famous block ciphers (including AES, MISTY1, Skipjack,...). Even if block ciphers have lots advantages compared with stream ciphers (they could be used for the both secure modes required for the sensor networks: the pairwise secure links and the secure group communications), the stream ciphers are usually used when wireless communications are required (as done in the WEP for example) because they could reach important flows for limited costs and the use of the "one time pad" encryption do not propagate errors induced by the communication channel. This cipher method combines using a modular addition (for example the XOR) the plaintext with a random key or "pad" used only once and having the same length than the plaintext. The pad also called pseudo-random sequence is produced using a pseudo-random generator or a synchronous stream cipher and is generated from the secret shared key K and an initial value IV, that must be different for each encryption. So, in the context of sensor networks, stream ciphers could be useful for pairwise secure associations.

This article then proposes to theoretically sum-up the security provided by some stream ciphers dedicated to software uses (especially the ones recently proposed in the European Project Ecrypt, workpackage eStream [27]) and presents some implementation tests performed on a dedicated platform of sensors [16].

This paper is organized as follows: Section 1 presents several stream ciphers and evaluates their current security based on the most recent results. Section 2 presents the dedicated platform and describes the methodology used to perform our benchmarks. Section 3 provides our results and our analysis concerning the benchmarking whereas Section 4 concludes this paper.

1 The Studied Stream Ciphers

We decided to study and to benchmark the stream ciphers dedicated to software uses (Profile 1 of the eStream call for primitives) submitted to the eStream call and belonging to the Focus Phase 2 (see the website of the eStream project `http://www.ecrypt.eu.org/stream/phase2list.html` for more details of their choice). This project is ongoing so the security evaluation of the proposed stream ciphers is always in hand. Even if the security study concerning these primitives is not finished, it seems interesting for us to study their performances on a dedicated architecture with strong constraints due to their high efficiency and their high reliability in wireless context. Moreover, in most of cases, the code size required for stream ciphers is smaller than the one required for block ciphers.

We have added to those ciphers three other pseudo-random generators due to their fame and their great use: RC4 (always used in WPA and in `https`), SNOW v2 [12] (the updated version of the NESSIE call for primitives [26]) and AES-CTR (the block cipher AES used in a particular mode, the CTR one) used in WPA2. Moreover, the AES-CTR is in fact a modified block cipher and then its performances correspond to those of a block cipher. We could then compare the results obtained for it as a block cipher to those of stream ciphers.

All the stream ciphers presented in this section uses at least the following parameters as suggested in the initial call of eStream: a secret shared key K of at least 128 bits and an IV value of at least 64 bits, that must be absolutely different at each new encryption.

In our security analysis, we claim that a stream cipher is secure until now if no attack (with a complexity less than 2^{128} or respecting the recommendations of the authors) has been exhibited against it until now.

RC4. RC4 was introduced in 1987 by R. Rivest [30] for the RSA laboratories. RC4 is most commonly used to protect Internet traffic using the SSL (Secure Sockets Layer) protocol. It is composed of an initialization phase that transforms the secret shared key K of length between 40 and 1024 bits into an initial S permutation from $N = 2^n$ into itself (typically $n = 8$). The stream sequence $z(t)$ is then produced by outputting particular values of the S permutation updated at each clock.

Security. In despite of many efforts provided by the cryptographers to try to break RC4, very few attacks are known against it. The strangest remains the "Finney property" (see [23] for more details). However, some statistical bias could be exhibited (see [23]) that allow to construct distinguishing attacks against RC4. An other attack proposed by S. Fulher, I. Mantin et A. Shamir [15] exploits the bad re-synchronization of RC4 and could be applied in the WEP case (see [25] and [24]).

RC4 stays a secure cipher for good initial choices: if the key scheduling algorithm is strengthened by pre-processing the base key (of at least 128 bits) and any counter or initialization vector by passing them through a hash function such as MD5 or by discarding the first 256 output bytes of the pseudo-random generator before beginning encryption (as described in [21]). Using those recommendations, we say that RC4 is secure.

SNOWv2. SNOW is a stream cipher submitted by P. Ekdahl and T. Johansson to the NESSIE call for primitives [11]. Several attacks have been exhibited against the first SNOW version ([17] and [8]) and thus obliged the authors to modify their initial submission. That has been done and a second version of SNOW, SNOW v2, have been proposed [12]. In this version, the secret shared key has a length of 128 or 256 bits whereas the use of an IV value of 128 bits is optional. This stream cipher uses an LFSR of length 16 on $GF(2^{32})$ and a non linear finite state machine called FSM. The first 32 bits output is generated after 32 clocks.

Security. The only attack describes against SNOW v2 has been proposed in [32] and requires 2^{225} output words (2^{230} bits) and 2^{225} steps of analysis to distinguish the output of SNOW 2.0 from a truly random bit sequence. This attack does not really endanger the security of SNOW because it just allows to distinguish the output sequence from a perfect random one and the required complexity is not currently reachable. So, we say that SNOW v2 is secure.

AES-CTR. The AES-CTR is not exactly a stream cipher (see for example [18] for more details). In fact, it uses the AES block cipher (see [14] for further details) in a particular mode of operation. The block cipher AES uses a key of length 128, 192 or 256 bits and encrypts using a parallel structure blocks of size 128 bits. The CTR mode of operations consists in ciphering a counter value - that must be used only one time for a given key as mentioned in the chapter 2 of [18] - with a particular key K and x-oring the ciphertext obtained with the corresponding block of plaintext. The counter corresponding with a IV value is then updated to cipher in “one-time pad” mode the next plaintext block.

Security. The AES block cipher has been chosen in 2001 as the new block cipher standard by the NIST after 4 years of study. So, we say that this block cipher (used in all the known modes of operation) is secure. Moreover, the study of this block cipher allows us to compare the performances of it as a block cipher with the other stream ciphers.

DRAGON. DRAGON [10] was submitted to the eStream call for primitives and is one of the FOCUS Phase 2 stream ciphers [27]. It is left unchanged compared to Phase 1, the initial phase of evaluation of the eStream project. Two versions have been proposed: Dragon-256 that uses a secret master key of 256 bits, and a publicly known initialization vector (IV), also of 256 bits; and Dragon-128 that uses 128-bit key and IV. The two versions uses a non-linear feedback shift register (NLFSR) of length 1024 bits and a nonlinear filter function from $\{0,1\}^{192}$ into itself with a 64-bit memory component.

Security. In [13], Englund and Maximov describe a distinguishing attack against Dragon-256, under the assumption that the cryptanalyst can obtain an enormous amount of keystream from a single key-IV pair. Both variants of the distinguishing attack require 2^{155} words of keystream with an operational complexity of 2^{187} and uses 2^{32} words of memory for the first variant and with a complexity of 2^{155} in time and of 2^{96} in memory for the second variant. However, those attacks do not take into account the authors recommendations: “ To protect against unknown future attacks, and against attacks that require large amounts of keystream, [Dragon] should be rekeyed at least once for every 2^{64} bits of keystream generated”.

In [7], an other statistical bias has been exhibited (with a probability equal to $2^{-92.8}$). But to detect this bias the amount of keystream required for the attack is by far larger than the limit of keystream available from a single key. So, until now, we say that Dragon is secure.

HC-256 and HC-128. Two versions of HC have been proposed in [34] and in [35]. The first one HC-256 generates keystream from a 256-bit secret key and a 256-bit initialization vector whereas the second one HC-128 supports 128-bit key and 128-bit initialization vector but only 2^{64} keystream bits can be generated from each key/IV pair. The general principle of the keystream generation for HC-256 is as follows: at each clock, a 32-bit word of one of the two secret tables (initialized with the key K and the IV value) is updated using a non-linear feedback function. Each table contains 1024 32-bit words. Every 2048 steps all the elements of the two tables are updated. At each step, HC-256 generates one 32-bit output using a 32-bit-to-32-bit mapping. HC-128 is the simplified version of HC-256: it uses two secret tables, each one having 512 32-bit elements. At each clock, one element of a table is updated using a non-linear feedback function. All the elements of the two tables are updated every 1024 clocks. At each clock, one 32-bit output is generated from the non-linear output filtering function.

Security. Until now, no attack have been found against HC-256 and HC-128. So we say that at this moment, the two HC versions are secure.

LEX. The stream cipher LEX has been proposed by A. Biryukov [5] and has been tweaked to enter Phase 2 of estream. This new version extracts parts of the internal state at certain rounds of the block cipher AES. The AES usual key lengths could be used: 128, 192 or 256 bits. The size of the IV is 128 bits. The output sequence is generated by outputting at each AES round certain four bytes from the intermediate variables. The difference with AES is that the attacker never sees the full 128-bit ciphertext but only portions of the intermediate states.

Security. The first version of LEX was successfully attacked by Hongjun Wu and Bart Preneel in [36] leading to a modified IV injection as done in the second version of LEX. Until now, no attack have been found against this new version. So we say that at this moment, LEX is secure.

Phelix. Phelix is a stream cipher proposed by D. Whiting, B. Schneier, S. Lucks and F. Muller [33]. It uses a 256-bit key and a 128-bit IV value. It has an internal state that consists of nine words of 32 bits each. The state is broken up into two groups: 5 "active" state words, which participate in the block update function, and 4 "old" state words that are only used in the keystream output function. Twenty elementary rounds are applied to produce one 32-bits output block.

Security. A very recent attack has been proposed by Hongjun Wu and Bart Preneel against Phelix in [37]. This attack is a differential-linear one assuming nonce reuse (corresponding with a chosen nonce attack). In this context, with 2^{34} chosen nonces and 2^{37} chosen plaintext words, the key of Phelix can be recovered with about $2^{41.5}$ operations. Even if this kind of attacks is not clearly authorized by the cryptographic community, it directly asks the question of the security of Phelix.

Py and Pypy. The stream cipher Py has been proposed by E. Biham et J. Seberry in [3]. It uses the same principles of construction than RC4 on two larger tables with a rolling update under keys of length up to 256 bits and IV of length 128 bits. At each clock, 64 bits of the output sequence are produced.

The allowed stream size is 2^{64} bytes for each stream sequence. For the eStream phase 2, an other stream cipher called Pypy (see [4]) has been proposed that outputs every second word of Py (only 32 bits are outputted at each clock).
Security Many attacks have been proposed against Py and Pypy: the first one [28] (improved in [9]) do not really endanger the security of Py and Pypy because they use more than 2^{64} bytes for each stream sequence. More recently, an other series of attacks using chosen IVs to recover the secret key has been proposed in [38] and in [19]. Those attacks seem to be more devastator than the previous ones. Then the security of Py and Pypy must be more carefully studied during the second eStream Phase.

Salsa20. The stream cipher Salsa20 has been proposed by D.J. Bernstein in [2]. It uses a key with a length from 16-byte to 32-byte and an *IV* of length 16-byte. The core of Salsa20 is a hash function with 64-byte input and 64-byte output. The hash function is used in counter mode as a stream cipher: Salsa20 encrypts a 64-byte block of plaintext by hashing the key, nonce, and block number and xor'ing the result with the plaintext.

Security. Until now, no attack has been found against Salsa20. So we say that at this moment, Salsa20 is secure.

SOSEMANUK. The stream cipher SOSEMANUK has been proposed by C. Berbain *et al.* in [1]. Its key length is variable between 128 and 256 bits. It accommodates a 128-bit initial value. Any key length is claimed to achieve 128-bit security. The SOSEMANUK cipher uses both some basic design principles from the stream cipher SNOW 2.0 and some transformations derived from the block cipher SERPENT. Sosemanuk aims at improving SNOW 2.0 both from the security and from the efficiency points of view.

Security. Until now, no attack with a complexity less than 2^{128} has been found against SOSEMANUK. So we say that at this moment, SOSEMANUK is secure.

2 Methodology

In this section, we present the platform used to perform the benchmarks and we also describe the testing framework.

2.1 The Dedicated Platform

All the benchmarks performed here are produced using a sensor platform built upon an ARM9 processor. Its processing power and the current evolution in processor size and energy consumption make it a rather good representative for next generation sensor network nodes. Nowadays the ARM7, which used to be a full featured processor, is considered as a 32 bit micro-controller, for example embedded in nearly all Bluetooth devices and in some wireless devices. Today, current sensor network data, like temperatures, require only few processing on the nodes, but we can state that next generation sensors will capture sounds or even images which will need more powerful nodes. We then decide to use an ARM9 core based CPU architecture for its computing power.

The platform is an ARM based development board. It uses an ARM922T, more precisely an Altera Excalibur EPXA10, which is a FPGA integrating an ARM922T core and usual embedded systems peripherals (*e.g.* UART, Timers) on the same chip. The processor accesses all peripherals and memory levels through two levels of AMBA bus. Memories are organized in a three level hierarchy. First level, the nearest from the processor are caches, two 8 kB of separated cache. At the second level we can find two 256kB and two 128kB scratch pad memories not used in the benchmarks. Finally, main memory, the furthest from processor, is a 128 MB SDRAM.

As far as benchmark construction is concerned, they was compiled with the standard GCC C compiler targeted to ARM processors. For the `libc` and operating systems functionalities, we used a lightweight operating system called `Mutek` [29]. It is Posix threads capable, but for the sack of predictability, all benchmarks are mono-threaded.

The energy and time performance informations are collected thanks to a two step simulation. The first step is the full architecture simulation. The simulator we use for this step is derived from the open source `skyeye` [31] simulator. Skyeye is a functionnal simulator targeted to ARM based embedded systems. Several full platforms are available for simulation like full featured PDA. This simulator is augmented in our case for our CM922T-XA10 platform support and we also added instruction cycle accuracy timing and peripheral activity reporting. This simulator is responsible for generating the linear execution tracelater used by the second simulator `esimu`. `esimu` generates a full profile in terms of time and energy of each benchmark. The results can then be visualized with profiles visualization tools freely available like `KCacheGrind` [20].

2.2 Methodology

We have adopted the methodology provided with the eStream testing framework (see [6] for more details) because it seems to be the most relevant one to evaluate stream ciphers. Indeed, a stream cipher is composed of an initial step, called the warm up phase, that produces from the key and the IV value an internal state that will produce the first output bits or bytes. We then need to test the time required to perform the “key setup” and the “IV setup”. Moreover, one of the main advantages of stream ciphers is that they are able to produce very quickly long sequences required for the ciphering operation. We then to measure this particular property.

The set of tests are performed in order to study the specific requirements on the efficiency of the primitives in various situations. The testing framework described in [6] then proposes four performance measures to test the most relevant implementation properties:

– **Encryption rate for long streams:** This aspect reflects the biggest potential advantage over block ciphers and appears as an important criterion in many applications. We have decided to measure here the encryption rate by ciphering a long stream in chunks of about 4Kb. The encryption speed is computed in cycles/byte by measuring the cycles required to encrypt 10 such blocks under 10 different keys. The time to setup the key and the *IV* is not considered in this test.

- **Packet encryption rate:** while a block cipher is likely to be a better choice when encrypting very short packets, it is interesting to determine at which length a stream cipher starts to take the lead. The packet encryption rate is measured in cycles/byte for three packet lengths (40, 576 and 1500 bytes) including an *IV* setup and a MAC finalization if an authenticated encryption is supported (only Phelix has this property). This test is repeated under 10 different keys on several packets.
- **Key and *IV* setup:** The last test separately measures the efficiency of the key setup and of the *IV* setup. "This is probably the least critical of the four tests, considering that the efficiency of the *IV* setup is already reflected in the packet encryption rate, and that the time for the key setup will typically be negligible compared to the work needed to generate and exchange the key." ([6]). The tests are performed for several key and *IV* values and the results are provided in cycles/key or cycles/*IV*.
- **Agility:** When an application needs to encrypt many streams in parallel on a single processor, its performance will not only depend on the encryption speed of the cipher, but also on the time spent switching from one session to another. The testing framework performs the following test: it first initiates a large number of sessions (filling 16MB of RAM), and then encrypts streams of plaintext in short blocks of around 256 bytes, each time jumping from one session to another. The results of this test are provided in cycles/byte repeating the test on 270 blocks of 256 bytes under one key.

We also perform some tests concerning the code size required to embed such ciphers on the platform. We refer to two types of memory: the code memory in the form of flash memory and the data memory in the form of RAM. We have performed those tests on the same kind of codes for each stream cipher including a key-setup, an *IV*-setup and a call to the function that encrypts long streams.

3 Results

To perform our benchmarks using the previous methodology, we have used without modifying them, the C codes provided in the testing framework [6]. All the C codes are available via the webpage `http://www.ecrypt.eu.org/stream/perf/`. To obtain a point of comparison, results are also given for a simple Copy operation, this code is also provided by the testing framework.

3.1 CPU Cycles and Energy Consumption

The results (computed using the `skyeye` + `eSimu` tools) concerning the number of cycles required to perform all the tests are summed up in the table 1.

Table 1. Number of CPU cycles for the stream ciphers using the testing framework

			cycles/byte				cycles/key	cycles/IV	cycles/byte
Algo.	Key	*IV*	Stream	40 bytes	576 bytes	1500 bytes	Key setup	IV setup	agility
Copy	80	80	2.19	3.72	1.00	7.58	4.40	4.19	7.78
RC4	128	0	26.97	610.95	58.53	33.29	76.41	23581.61	21.24
SNOW v2.0	128	128	25.08	66.38	16.82	23.71	163.41	2273.35	20.87
AES CTR	128	128	206.19	131.52	198.73	195.76	636.49	157.52	202.23
DRAGON	128	128	30.89	177.05	69.76	64.91	421.42	4497.61	33.60
HC-256	128	128	27.00	6044.76	446.11	183.17	141.75	198126.10	49.30
HC-128	128	128	19.35	1484.72	112.12	53.70	141.76	58194.93	31.67
LEX	128	128	47.07	71.41	40.32	41.92	501.41	1415.57	50.71
Phelix	128	128	25.61	90.15	28.36	26.77	1271.42	2154.61	26.99
Py	128	64	214.25	349.23	47.58	60.88	7713.83 z	9327.43	64.40
Pypy	128	56	44.78	360.95	103.91	74.72	7713.82	9660.11	73.46
Salsa20	128	64	57.54	84.57	55.05	73.07	367.70	118.07	72.60
SOSEMANUK	128	64	14.81	385.63	37.95	30.48	16374.01	1264.09	20.78

Table 2. Number of nJ for the stream ciphers using the testing framework

			nJ/byte				nJ/key	nJ/IV	nJ/byte
Algo.	Key	*IV*	Stream	40 bytes	576 bytes	1500 bytes	Key setup	IV setup	agility
Copy	80	80	38.32	60.85	16.84	142.07	70.54	67.29	145.35
RC4	128	0	465.17	9843.25	948.49	542.06	1243.66	379636.24	354.43
SNOW v2.0	128	128	438.34	1093.46	280.59	414.20	2656.66	41749.08	365.26
AES CTR	128	128	3587.00	2197.89	3437.36	3384.26	11378.81	2861.89	3499.45
DRAGON	128	128	514.26	2912.69	1144.53	1064.58	6846.80	74109.24	575.67
HC-256	128	128	471.39	102473.69	7577.28	3112.48	2540.02	2705307.85	864.11
HC-128	128	128	342.29	24264.78	1838.04	897.21	2540.20	950661.16	559.97
LEX	128	128	804.03	1186.80	670.42	714.16	8250.66	23850.60	868.13
Phelix	128	128	421.15	1470.51	461.14	454.71	20622.78	35111.32	461.26
Py	128	64	3894.22	5822.63	827.52	1101.62	145194.31	154181.03	1141.65
Pypy	128	56	817.35	6008.43	1859.92	1361.36	145194.15	161834.16	1300.67
Salsa20	128	64	952.19	1394.11	907.17	1275.82	6884.19	2215.93	1268.12
SOSEMANUK	128	64	247.93	6727.04	648.50	528.97	286119.29	20860.01	365.30

The results concerning the energy consumption are given in the table 2. The key and the *IV* sizes used to perform the tests are also specified.

3.2 Memory Requirements

We have performed some tests concerning the memory requirements of all the ciphers (using the same key and *IV* lengths) using only a speed option of optimization under `gcc` (the `-O2` one). The results concerning the code and the data memory sizes of all ciphers are given in table 3, together with the results for an empty code and always for the Copy in order to evaluate the minimum memory size induced by the benchmark environment.

Table 3. Code Memory Requirements and Data Memory Requirements in bytes and in decimal notations

Algo.	Empty	Copy	RC4	SNOW v2.0	AES-CTR	DRAGON	HC-256
Code size	4992	5040	6064	11152	17456	8512	14432
Data size	480	752	692	6836	13020	2740	692
Algo.	HC-128	LEX	Phelix	Py	Pypy	Salsa20	SOSEMANUK
Code size	12496	13072	9968	8736	8512	6560	21968
Data size	692	5852	724	35248	35248	724	3164

3.3 Analysis

First, we could see that most of stream ciphers (SNOW v2.0, SOSEMANUK, Dragon,...) stay more efficients on the dedicated architecture than the AES block cipher used in the CTR mode if we do not take into account the time required for the key setup and for the *IV* setup. Some of them such as Salsa20 have also a more efficient key and *IV* setup. Moreover, the code memory size and the data memory size of the AES-CTR is among biggest (except for Py and Pypy for the data memory size and for SOSEMANUK concerning the code memory size). So using stream ciphers in sensor network applications could be a good solution to achieve high encryption speed in high constraint environments.

We have then compare our benchmarks obtained on the dedicated platform and the results provided on the page of the eStream testing framework (see [6] for more details) that are given for traditional architectures such as Intel Pentium 4, Power PC,... (using several compilers). A really interesting point is that the most reliable stream ciphers on traditional platforms, Py and Pypy, do no longer work rapidly on our platform whereas SNOW v2.0, SOSEMANUK or HC-128 stay relatively fast.

This unusual property comes from the intrinsic structure of the ciphers Py and Pypy: they both uses two rolling tables of 256 bytes. The use of many `memcpy` to build at each iteration those tables explains the bad results obtained: the number of DC-misses is huge compared with the other studied stream ciphers. On our platform, and this is the case on other current sensor nodes, when the CPU accesses memory, it is stalled since it is not superscalar as many high performance architectures are (the Pentium architecture for example). Our platform embed caches, but their size are about the quarters of level 1 caches of current high performance processors and it has no level 2 cache. Thus this cache architecture is not compatible with the memory access hunger of Py and Pypy algorithms, even if the results obtained for Pypy are rather better. The data are not preserved in data cache at each access and need to be refetched from main memory. This characteristic was underlined by table 3 with the data segment size of the implementation of these two algorithms. In summary, whereas Py and Pypy are really efficient on traditional high performance architectures, they could not be used without modifications in a such constrained environment. Moreover, the structure of Pypy is the same than the one of Py and the results obtained for Pypy stay reasonable so we think that because the Pypy performances are more reasonable, Py is the only algorithm that is totally incompatible with this cache geometry.

An other surprising result concerns the number of cycles required by the Key-setup of SOSEMANUK that is very huge compared with the results obtained on the other traditional platforms. This very bad result could be explained by the excessive code size (shown by table 3) that involves in our highly constrained architecture a performances reduction. When we are looking at the SOSEMANUK C source, we could notice that loops were unrolled for performance purposes. This code is in fact optimized for more powerful architecture. But this improvement induce the same behavior in the instruction cache than Py and PyPy in the data cache. Cache often misses and instruction are fetched from memory while the CPU is stalled. Then, if we want to improve the performances of SOSEMANUK, a good solution seems to be to reduce the code size to make it fit in the cache.

The other observed results here go in the same direction than the one presented in [6]: the most rapid algorithms stay approximatively the same (except for the particular case of Py and Pypy): HC-128, SOSEMANUK, SNOW v2.0, Phelix, RC4 and HC-256. We do not have modified the C code provided on the eStream web page but we think that it could be a solution to improve the results of some algorithms if we use a lower level programming language.

4 Conclusion

We have presented here some benchmarks performed on stream ciphers, the traditional ones (RC4, SNOW v2.0,...) and the candidates of the ECRYPT project. Some results could appear very strange but are in fact conditioned by the physical constraints of our platform.

Due to the ongoing state of the stream ciphers studied here, we do not have to give any recommendation about their use in such constraint environment but in the case of well-known and well-studied stream ciphers, we could notice that SNOW v2.0 is swift as well on traditional platforms as on the highly constrained environment.

As part of future work, we will benchmark the same ciphers on a MS430 16 bit micro-controller. Then, the comparison between the results obtained in [22] concerning the performances of block ciphers using several modes of operation and the stream ciphers presented here will be more pertinent. We also want to estimate the general loose of performance produced by the addition of a stream cipher in a real sensor communication environment.

References

1. C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. SOSEMANUK: a fast oriented software-oriented stream cipher. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2005. `http://www.ecrypt.eu.org/stream/`.

2. Daniel J. Bernstein. Salsa20 specification. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2005. http://www.ecrypt.eu.org/stream/.
3. Eli Biham and Jennifer Seberry. Py: A fast and secure stream cipher using rolling arrays. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2005. http://www.ecrypt.eu.org/stream/.
4. Eli Biham and Jennifer Seberry. Pypy: Another version of py. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2006. http://www.ecrypt.eu.org/stream/.
5. Alex Biryukov. A new 128-bit key stream cipher lex. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2006. http://www.ecrypt.eu.org/stream/.
6. C. De Cannière. estream optimized code HOWTO. eSTREAM, ECRYPT Stream Cipher Project, 2005. http://www.ecrypt.eu.org/stream/perf/.
7. Joo Yeon Cho. An observation on dragon. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/002, 2007. http://www.ecrypt.eu.org/stream.
8. Don Coppersmith, Shai Halevi, and Charanjit S. Jutla. Cryptanalysis of stream ciphers with linear masking. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2002.
9. Paul Crowley. Improved cryptanalysis of py. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/010, 2006. http://www.ecrypt.eu.org/stream.
10. Ed Dawson, Kevin Chen, Matt Henricksen, William Millan, Leonie Simpson, Hoon-Jae Lee, and SangJae Moon. Dragon: A fast word based stream cipher. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2005. http://www.ecrypt.eu.org/stream/.
11. P. Ekdahl and T. Johansson. SNOW - a new stream cipher. In *Proceedings of First NESSIE Workshop*, Heverlee, Belgique, 2000.
12. P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In *Selected Areas in Cryptography – SAC 2002*, volume 2295 of *Lecture Notes in Computer Science*, pages 47–61. Springer-Verlag, 2002.
13. Hakan Englund and Alexander Maximov. Attack the dragon. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/062, 2005. http://www.ecrypt.eu.org/stream.
14. FIPS 197. Advanced Encryption Standard. Federal Information Processing Standards Publication 197, 2001. U.S. Department of Commerce/N.I.S.T.
15. S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of rc4. In *Selected Areas in Cryptography - SAC 2001*, volume 2259 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, 2001.
16. N. Fournel, A. Fraboulet, and P. Feautrier. Booting and Porting Linux and uClinux on a new platform. Research Report RR2006-08, LIP - ENS Lyon, Feb 2006.
17. P. Hawkes and G. Rose. Guess-and-determine attacks on SNOW. In *Selected Areas in Cryptography - SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 37–46. Springer-Verlag, 2002.
18. R. Housley. Using advanced encryption standard (aes) counter mode with ipsec encapsulating security payload (esp). IETF, RFC 3686, 2004. http://www.rfc-archive.org/getrfc.php?rfc=3686.
19. Takanori Isobe, Toshihiro Ohigashi, Hidenori Kuwakado, and Masakatu Morii. How to break py and pypy by a chosen-iv attack. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/060, 2006. http://www.ecrypt.eu.org/stream.
20. Available online, http://kcachegrind.sourceforge.net/, nov 2006.

21. RSA laboratories. Rsa security response to weaknesses in key scheduling algorithm of rc4. available at `http://www.rsasecurity.com/rsalabs/node.asp?id=2009`, 2007.
22. Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):65–93, 2006.
23. I. Manti and A. Shamir. A practical attack on broadcast rc4. In *Fast Software Encryption - FSE 2001*, volume 2335 of *Lecture Notes in Computer Science*, pages 152–164. Springer-Verlag, 2001.
24. Itsik Mantin. A practical attack on the fixed rc4 in the wep mode. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 395–411. Springer, 2005.
25. Itsik Mantin. Predicting and distinguishing attacks on rc4 keystream generator. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2005.
26. NESSIE. Nessie phase 1 : selection of primitives. `https://www.cryptonessie.org/`, 2001.
27. Network of Excellence in Cryptology ECRYPT. Call for stream cipher primitives. `http://www.ecrypt.eu.org/stream/`.
28. Souradyuti Paul, Bart Preneel, and Gautham Sekar. Distinguishing attacks on the stream cipher py. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2006.
29. Frédéric Pétrot and Pascal Gomez. Lightweight Implementation of the POSIX Threads API for an On-Chip MIPS Multiprocessor with VCI Interconnect. In *DATE 03 Embedded Software Forum*, pages 51–56, 2003.
30. R. Rivest. The RC4 encryption algorithm. RSA Data Security, 1992.
31. Available online, `http://www.skyeye.org/`http://www.skyeye.org/, nov 2006.
32. D. Watanabe, A. Biryukov, and C. De Cannire. A distinguishing attack of SNOW 2.0 with linear masking method. In *Selected Areas in Cryptography 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 222–233. Springer-Verlag, 2003.
33. Doug Whiting, Bruce Schneier, Stephan Lucks, and Frédéric Muller. Phelix - fast encryption and authentication in a single cryptographic primitive. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2005. `http://www.ecrypt.eu.org/stream/`.
34. Hongjun Wu. Stream cipher hc-256. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2005. `http://www.ecrypt.eu.org/stream/`.
35. Hongjun Wu. Stream cipher hc-128. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2006. `http://www.ecrypt.eu.org/stream/`.
36. Hongjun Wu and Bart Preneel. Attacking the iv setup of stream cipher lex. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/059, 2005. `http://www.`ecrypt.eu.org/stream.
37. Hongjun Wu and Bart Preneel. Differential-linear attacks against the stream cipher phelix. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/056, 2006. `http://www.ecrypt.eu.org/stream`.
38. Hongjun Wu and Bart Preneel. Key recovery attack on py and pypy with chosen ivs. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/052, 2006. `http://www.ecrypt.eu.org/stream`.

Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures

Chong Hee Kim and Jean-Jacques Quisquater

UCL Crypto Group, Université Catholique de Louvain, Belgium
kim@dice.ucl.ac.be, quisquater@dice.ucl.ac.be

Abstract. Nowadays RSA using *Chinese Remainder Theorem* (CRT) is widely used in practical applications. However there is a very powerful attack against it with a fault injection during one of its exponentiations. Many countermeasures were proposed but almost all of them are proven to be insecure. In 2005, two new countermeasures were proposed. However they still have a weakness. The final signature is stored in a memory after CRT combination and there is an error-check routine just after CRT combination. Therefore, if an attacker can do a double-fault attack that gives the first fault during one of the exponentiation and the other to skip the error-checking routine, then he can succeed in breaking RSA. In this paper, we show this can be done with the concrete result employing a glitch attack and propose a simple and almost cost-free method to defeat it.

1 Introduction

After the advent of the concept of *Side Channel Analysis* by Kocher [15], many variants have appeared to attack the embedded systems such as smart cards. Among them, fault attacks introduced by Boneh et al. [5] are the most effective.

There are several kinds of methods to invoke faults such as variations in supply voltage, variations in the external clock, temperature variation, white light, laser, and X-rays and ion beams [3]. The objective of invoking faults is to make an abnormal operation in a target and to compute hidden secret information with the faulty output. In this paper, we use a glitch attack which makes a transient fault with a voltage spike. The target of glitches is to corrupt data transferred between registers and memory or to prevent the execution of the code. The glitch attack is used to attack RSA [1] and recently DSA [16]

The first victim of the fault attack on the cryptographic algorithms was RSA. The straightforward RSA implementation with Chinese Remainder Theorm was shown to be broken by fault attacks [5]. The simplest way to prevent the fault attack is just to compute signatures twice and compare them. However this doubles computation time. Furthermore it cannot avoid permanent errors. Another way is to verify the signature with the public exponent e. That is, the device

D. Sauveron et al. (Eds.): WISTP 2007, LNCS 4462, pp. 215–228, 2007.
© IFIP International Federation for Information Processing 2007

returns the signature S only when $S^e \equiv m \pmod{N}$. However this method is too costly if e is large. Furthermore in some applications (e.g. javacard), it is not allowed to access the public exponent e during signature generation.

Many countermeasures were proposed [17,1,21] but they have been broken [13,18]. In 2005, two new algorithms were proposed by Ciet and Joye [8] and Giraud [10] separately. However, their schemes also missed one important point. In the next section, we review the RSA-CRT and the existing countermeasures against fault attacks. Section 3 shows the problem of the existing countermeasures and experimental results. Section 4 present the new approach to avoid the new attack and finally we conclude in Section 5.

2 Previous Countermeasures

In this section, we briefly review the RSA-CRT (CRT based RSA) signature and countermeasures against fault attacks .

2.1 RSA-CRT Signature Algorithm and Fault Attacks on It

Let $N = p \cdot q$ be the RSA modulus, where p and q are two large primes. Let e be the RSA public exponent and d be the RSA private exponent satisfying that $e \cdot d = 1 \bmod (p-1)(q-1)$. We also let d_p (resp. d_q) be the CRT exponent such that $d_p = d \bmod (p-1)$ (resp. $d_q = d \bmod (q-1)$). We denote by Iq the inverse of q modulo p. Then the signature S of the message m is computed as

1. $S_p = m^{d_p} \bmod p$
   $S_q = m^{d_q} \bmod q$

2. $S = \mathrm{CRT}(S_p, S_q) = S_q + q \cdot ((S_p - S_q) \cdot Iq \bmod p)$.

Bellcore researchers showed that if an error occurs in only one of the exponentiations (that is, during a computation of S_p or S_q, but not in both), then the factorization of N is possible with the faulty signature $\tilde{S}$ [5]. For example, suppose that an error occurs during computation of S_p. Then a faulty $\tilde{S_p}$ will be used in a CRT combination and $\tilde{S} = \mathrm{CRT}\ (\tilde{S_p}, S_q)$ will be returned. With the correct signature S and the faulty one $\tilde{S}$, the secret prime q can be computed by computing GCD(S-$\tilde{S}$, N).

This attack is further improved with only one execution of algorithm [12]. Secret prime number q can be found by computing $\mathrm{GCD}(\tilde{S}^e - m, N)$.

2.2 Shamir's Countermeasure and Its Generalizations

Shamir used a redundant way to compute S_p and S_q and checked the correctness of S_p and S_q before RSA combination[17]. Let r be a random k-bit integer (typically, k=32). Then the signature is computed as

1. $S_p^* = m^d \bmod (p \cdot r)$
   $S_q^* = m^d \bmod (q \cdot r)$

2. $\begin{cases} S = \mathrm{CRT}(S_p^*, S_q^*) \bmod N \text{ if } S_p^* \equiv S_q^* \pmod{r}, \\ error \quad \text{otherwise.} \end{cases}$

Joye et al. pointed out one drawback of Shamir's method [13]. It requires d which is not known in CRT. Only $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$ are known. They proposed an improved algorithm which verifies the two half exponentiations separately. Let r_1 and r_2 be two random k-bit integers. Then device computes

1. $S_p^* = m^{d_p} \bmod (p \cdot r_1)$, $s_1 = m^{d_p \bmod \varphi(r_1)} \bmod r_1$
   $S_q^* = m^{d_q} \bmod (q \cdot r_2)$, $s_2 = m^{d_q \bmod \varphi(r_2)} \bmod r_2$

2. $\begin{cases} S = \mathrm{CRT}(S_p^*, S_q^*) \bmod N \text{ if } S_p^* \equiv s_1 \bmod r_1 \text{ and } S_q^* \equiv s_2 \bmod r_2 \\ error \quad \text{otherwise.} \end{cases}$

The previous algorithms cannot detect errors occurred during RSA combination. In [1], Aumüller et al. checked the correctness of the result of RSA combination. Let r be a short prime number, e.g., 16 bits. Then device computes

```
1. $p' = p \cdot r$
   $d'_p = d_p + random_1 \cdot (p-1)$
   $S'_p = m^{d'_p} \bmod p'$
   if $\neg(p' \equiv 0 \pmod{p} \wedge d'_p \equiv d_p \pmod{(p-1)})$ then return $error$

   $q' = q \cdot r$
   $d'_q = d_q + random_2 \cdot (q-1)$
   $S'_q = m^{d'_q} \bmod q'$
   if $\neg(q' \equiv 0 \pmod{q} \wedge d'_q \equiv d_q \pmod{(q-1)})$ then return $error$

2. $S_p = S'_p \bmod p$
   $S_q = S'_q \bmod q$
   $S = \mathrm{CRT}(S_p, S_q)$

3. if $\neg(S \equiv S_p \pmod{p} \wedge S \equiv S_q \pmod{q})$ then return $error$

   $S_{pr} = S'_p \bmod r$
   $d_{pr} = d'_p \bmod (r-1)$
   $S_{qr} = S'_q \bmod r$
   $d_{qr} = d'_q \bmod (r-1)$
   if $(S_{pr}^{d_{qr}} \equiv S_{qr}^{d_{pr}})$ then
       return $S$,
   else
       return $error$.
```

2.3 Infective Computations

Yen et al. proposed a different kind of approach, *fault infective computation* [21]. They noted that an error detection based on decisional tests should be avoided. From the viewpoint of low-level implementation of this decision procedure, it often totally relies on the status of the *zero flag* of a processor. The zero flag is a bit of the status register in a processor. So, if an attack can induce a random fault into the status register, then conditional jump instruction may perform falsely. In their method, if an error occurs in one of the exponentiations (S_p or S_q), then it makes both $\tilde{S} \not\equiv S \pmod{p}$ and $\tilde{S} \not\equiv S \pmod{q}$. Unfortunately, both their countermeasures were shown to be insecure by Yen and Kim [19].

Blömer et al. suggested another countermeasure based on Shamir's method and on fault infective computation [7]. Given a security parameter k, for two appropriately chosen k-bit integers r_1 and r_2 (stored in memory), the following quantities are pre-computed and stored in memory:

$$r_1 p, \quad r_2 q, \quad r_1 r_2 N,$$

$$\begin{aligned} d_1 &= d \bmod \varphi(r_1 p), \quad e_1 = {d_1}^{-1} \bmod \varphi(r_1), \\ d_2 &= d \bmod \varphi(r_2 q), \quad e_2 = {d_2}^{-1} \bmod \varphi(r_2). \end{aligned}$$

The device then computes

1. $S_p^* = m^{d_1} \bmod (r_1 p)$,
   $S_q^* = m^{d_2} \bmod (r_2 q)$,
2. $S^* = \mathrm{CRT}(S_p^*, S_q^*) \bmod (r_1 r_2 N)$,
3. $c_1 = (m - {S^*}^{e_1} + 1) \bmod r_1$,
   $c_2 = (m - {S^*}^{e_2} + 1) \bmod r_2$,
   $S = (S^*)^{c_1 c_2} \bmod N$.

If there is no error, then c_1 and c_2 become 1 and the device returns a correct signature S.

Unfortunately, this countermeasure is also shown to be insecure by Wagner [18]. Let us suppose a random transient fault that modifies the value of m as it is being read from memory in the computation of S_p^* while leaving the value stored in memory unaffected, then $c_1 \neq 1$ but $c_2 = 1$. Then the attacker can mount a Bellcore-like attack by computing $\mathrm{GCD}(m^{c_1} - S^e, N)$ with the guess of c_1. In the scheme, $c_1 = (m - S^{e_1} + 1) \bmod r_1$. Since $S^{e_1} = \tilde{m}$ can be guessed in his fault attack model, the attack was possible. Recently Blömer et al. proposed a variant that overcome the weakness by randomizing the computation of c_i [6].

2.4 Ciet and Joye's Countermeasure

In 2005, Ciet and Joye generalized Shamir's countermeasure [13] and adapted fault infective computation [21] to avoid decisional tests [8]. For two co-prime k-bit integers r_1 and r_2 and l-bit integer r_3, we define

$$p^* = r_1 p,$$
$$q^* = r_2 q,$$
$$I_q^* = (q^*)^{-1} \bmod p^*.$$

Then device computes

1. $S_p^* = m^{d_p} \bmod p^*$ and $s_2 = m^{d_q \bmod \varphi(r_2)} \bmod r_2$,
   $S_q^* = m^{d_q} \bmod q^*$ and $s_1 = m^{d_p \bmod \varphi(r_1)} \bmod r_1$,
2. $S^* = S_q^* + q^* \cdot I_q^* \cdot (S_p^* - S_q^*) \bmod p^*$,
3. $c_1 = (S^* - s_1 + 1) \bmod r_1$
   $c_2 = (S^* - s_2 + 1) \bmod r_2$
   $\gamma = \lfloor (r_3 c_1 + (2^l - r_3) c_2)/2^l \rfloor$
   $S = (S^*)^\gamma \bmod N$

2.5 Giraud's Countermeasure

In 2005, Giraud [10] used the fact that the temporary variables (a_0, a_1) are of the form $(m^\alpha, m^{\alpha+1})$ in Joye and Yen's SPA-countermeasure [14]. Let $(d_{n-1}, \ldots, d_0)$ be the binary representation of d. Then a safe-error resistant exponentiation based on Montgomery Ladder of [14] is computed as following:

```
$a_0 \leftarrow 1$
$a_1 \leftarrow m$
for $i$ from $n-1$ to 0 do
    $a_{\bar{d}_i} \leftarrow a_{\bar{d}_i} \cdot a_{d_i} \bmod N$
    $a_{d_i} \leftarrow a_{d_i}^2 \bmod N$
return $a_0$.
```

To construct a SPA-FA(fault attack)-resistant CRT-RSA, he first proposed SPA-FA-resistant modular exponentiation (d is supposed to be odd):

```
$a_0 \leftarrow m$
$a_1 \leftarrow m^2 \bmod N$
for $i$ from $n-2$ to 1 do
    $a_{\bar{d}_i} \leftarrow a_{\bar{d}_i} \cdot a_{d_i} \bmod N$
    $a_{d_i} \leftarrow a_{d_i}^2 \bmod N$
$a_1 \leftarrow a_1 \cdot a_0 \bmod N$
$a_0 \leftarrow a_0^2 \bmod N$
if (Loop Counter $i$ not modified) & (Exponent $d$ not modified) then
    return $(a_0, a_1)$,
else
    return *error*.
```

Giraud used $k \cdot N$ instead of N, where k is a 32-bit random number in [11]. Here, we denote above algorithm as $(A, B) \leftarrow$ SPA-FA-EXP(m, d, N). Then the output (A, B) is $(m^{d-1} \bmod N,\ m^d \bmod N)$. Finally SPA and FA-resistant CRT-RSA algorithm is as follows:

```
1.  $(S_p^*, S_p) \leftarrow$ SPA-FA-EXP$(m, d_p, p)$
    $(S_q^*, S_q) \leftarrow$ SPA-FA-EXP$(m, d_q, q)$

2.  $S^* = \mathrm{CRT}(S_p^*, S_q^*)$
    $S = \mathrm{CRT}(S_p, S_q)$
    $S^* = m \cdot S^* \bmod (p \cdot q)$

3   if $S^* = S$ & (Parameters $p$ and $q$ not modified) then
        return $S$
    else
        return *error*
```

3 Problem of Previous Countermeasures

The previously known countermeasures to defeat fault attacks on RSA-CRT mostly consist of three parts. Firstly the device computes two exponentiation S_p^* and S_q^*. The computation of S_p^* (resp. S_q^*) is done by either straightforward computation like $S_p = m^{d_p} \bmod p$ (resp. $S_q = m^{d_q} \bmod q$) or inclusion of a kind of redundancy which will be used later to check errors. Secondly it combines two exponentiations to compute signature S^*.

The final step can be divided into two categories. The first one uses conditional check routine in which if an error does not occur then it outputs correct signature and if error occurs it gives predefined signal like "error has been detected" (e.g. Aumüller et al.[1], Giraud's [10], etc.). In the other method, instead of using the conditional check routine it gives a random value instead of a signature if error occurs (e.g. Infective computations[21,6], Ciet and Joye's [8], etc.).

Step 1. Computation of two exponentiation
- Compute S_p^* and S_q^*

Step 2. CRT combination
- Compute $S^* \leftarrow \mathrm{CRT}(S_p^*, S_q^*)$

Step 3. Fault detection
- Return $\begin{cases} S \leftarrow f(S^*) & \text{if there is no error,} \\ \perp & \text{otherwise.} \end{cases}$

Suppose that the attacker tries to skip "fault detection" routine (Step 3 in the above model) after CRT combination (Step 2). Then the attacker can get S^*. Furthermore S can be computed easily since $S = S^*$ in the conditional check routine approach and $S = S^* \bmod N$ in the other approach. Therefore we can consider the following attack scenario.

Our Fault Attack Model. The attacker tries to do a double-fault attack. He gives the fist fault during only one of the two exponentiations to corrupt its value. Then he gives the other fault during fault detection routine to skip some operations. If he succeed in doing a double-fault attack, then he can get the output of the CRT combination. That is, he gives a fault during Step 1. and gets the output of Step 2. by skipping some operations of Step 3. by faults in the above model.

Unfortunately all previous known countermeasures can be vulnerable to our fault attack scenario. Because all of them check the occurrence of errors after computation of a final signature.

3.1 Experiments of Our Attack

General Description. As seen in the previous section, RSA-CRT with a countermeasure against fault attacks is composed of three parts. The attacker tries to give two times faults. In the first trial, he tries to make errors during only one of the two exponentiations during Step 1. If the attacker can get the faulty signature as the output of Step 2, then he can compute the private keys. Therefore, he gives his second faults during Step 3 to skip some operations.

In the next section, as an example, we chose Ciet and Joye's countermeasure and implemented it. We gave a fault during the computation of S_p^*. Then we gave the next fault during the computation of $S = (S^*)^\gamma \bmod N$ and tried to skip it.

Results. We implemented 128-bit RSA-CRT with Ciet and Joye's countermeasure (We note it as *RSA-CRT with CJ*) [8] in an Atmel 8-bit AVR microcontroller, ATMega168 [2]. The program is implemented as follows:

```
Main() {
        ...
        Set I/O pin low
        Call subroutine RSA-CRT with CJ (as in 2.4)
        Set I/O pin high
        ...
}
```

The tools used to create the glitches and the target board can be seen in Fig.1. The chip is communicating with a computer via serial communication and the power consumption is monitored by an oscilloscope even though they are not shown in the figure. Fig.2 shows the I/O pin and power profiles. The x-axis represents time and y-axis represents voltage (for I/O profile) and the consumption of power (for power profile). The upper line represents the profile of I/O behavior. The lower profile shows the power profile. The *RSA-CRT with CJ* starts at the time block 0.4 and ends at the time block 6.8. In the figure the blocks

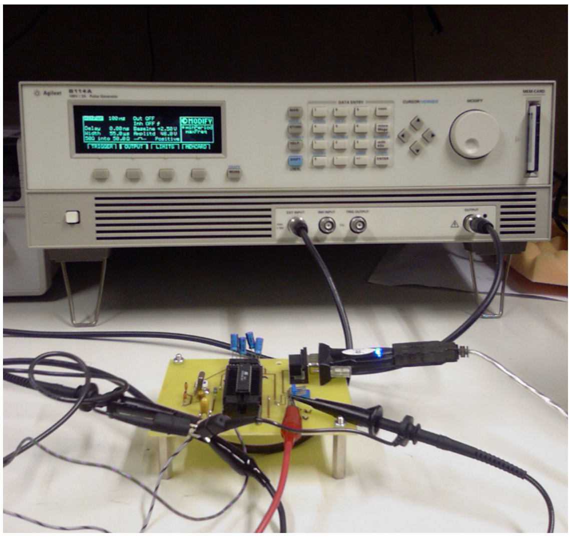


Fig. 1. Experiment setup for the glitch attack

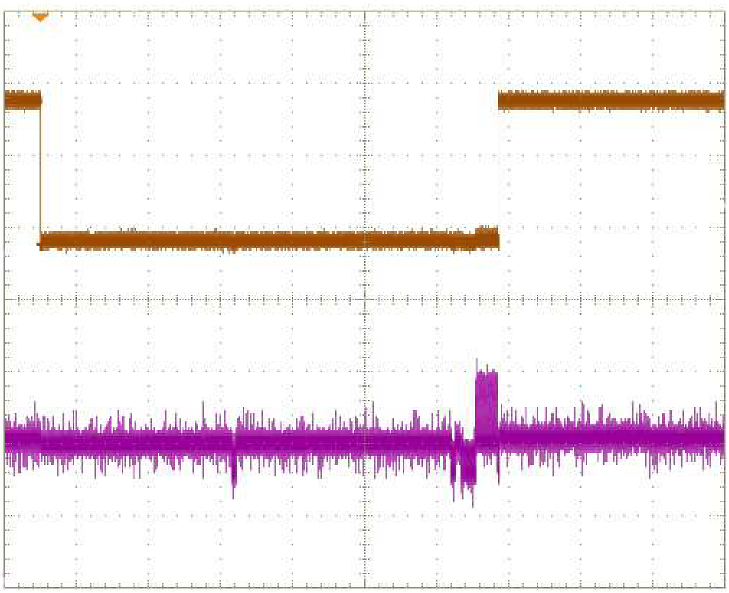


Fig. 2. RSA-CRT with CJ method without faults

are numbered from 0 to 10. In our case the first exponentiation S_p^* lies in the time frame 0.4 to 3.2. and the second one S_q^* in 3.2 to 6.0.

Firstly we gave a glitch into chip's power supply during the first exponentiation. You can see the result in Fig.3. There is a high peak in power profile in the time frame about 0.6. You can also see the increase of total execution time due to the faults. Since γ is no more 1, the computation of $S = (S^*)^\gamma \bmod N$ requires more time. In Fig.3 the time frame from 6.4 to 7.0 corresponds to this.

Then we gave another glitch just before the last computation of $S = (S^*)^\gamma \bmod N$ as in Fig.4. Compared to Fig.3, you can see the total time is reduced and the final computation of $S = (S^*)^\gamma \bmod N$ is disappeared. However you can see the chip is still working and PC(*program counter*) is in *main* function by seeing the I/O pin is *high*. If the chip is dead then the I/O pin will stay at *low* state. Because I/O pin is set to *high* in the main program after calling subroutine *RSA-CRT with CJ*. We confirmed the computation of $S = (S^*)^\gamma$ is skipped by reading the returned value and computing the prime number p and q with this fault signature with *Bellcore-like attack* [4].

In addition, the second fault skipping operations was more difficult than the one making faults during an exponentiation. Sometimes a chip was stunned and it never returned back to main function. Sometimes it showed whole RAM values (there is a command shows specific RAM value, but it seemed that there was a

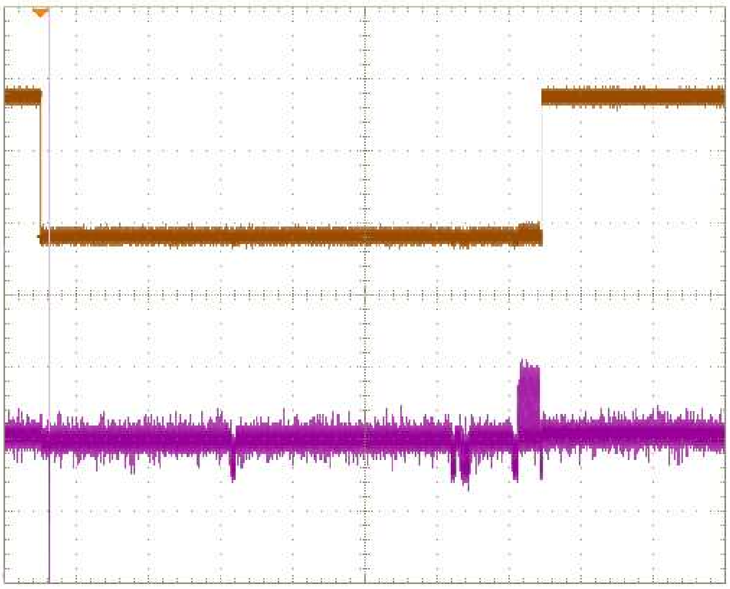


Fig. 3. Glitch attack during S_p^* operation

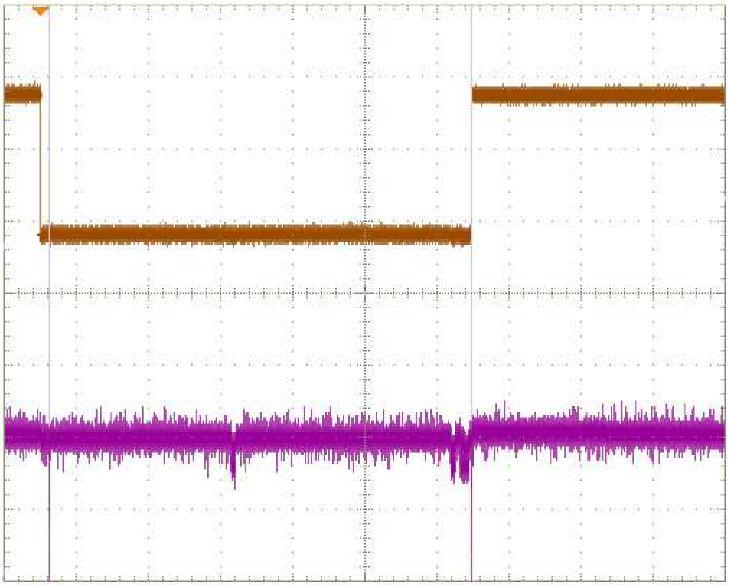


Fig. 4. Glitch attack both S_p^* operation and $(S^*)^\gamma \bmod N$

disturbance on the required address of RAM). We could also change the value of γ by giving a glitch during the computation of this.

4 New Approach to Prevent Fault attacks

The simplest method to prevent our attack in Ciet and Joye's scheme is to allocate all intermediate variables in different memory buffers that are not used for the returned signature. (Initialization of γ with a random value was not perfect in our experiment because we skipped only the last exponentiation after computation of γ). It means, for example, 128 bytes in a 1024-bit RSA implementation should not be used during whole RSA operations. It can be a burden for a programmer. Usually there is a specialized RAM, it is called *crypto RAM*, in a smart card only for cryptographic usage and this is not so large. Therefore the programmer should implement a RSA within comparatively small crypto RAM. Furthermore, because there is a possibility to know the value stored in RAM according to our experiments, the best idea is not to store the final signature until the end of checking errors.

Therefore we suggest more general idea to overcome previously mentioned problem. It is that the final signature S to be computed and stored only after Step 3. Then with the result of Step 2 the attacker could not get any useful information about secret keys. We modify Ciet and Joye's scheme and Giraud's scheme as examples. We put a randomness before CRT combination and get rid of it after an error check. Our idea can be applied for other countermeasures.

```
Modified Ciet and Joye's scheme

0. Choose a random integer $\mathbf{a}$ in $Z^*_{r_1 r_2 N}$
   Initialize $\gamma$ with a random number

1. $S_p^* = (\mathbf{a} + m^{d_p}) \bmod p^*$ and $s_2 = (\mathbf{a} + m^{d_q \bmod \varphi(r_2)}) \bmod r_2$,
   $S_q^* = (\mathbf{a} + m^{d_q}) \bmod q^*$ and $s_1 = (\mathbf{a} + m^{d_p \bmod \varphi(r_1)}) \bmod r_1$,

2. $S^* = S_q^* + q^* \cdot I_q^* \cdot (S_p^* - S_q^*) \bmod p^*$,

3. $c_1 = (S^* - s_1 + 1) \bmod r_1$
   $c_2 = (S^* - s_2 + 1) \bmod r_2$
   $\gamma = \lfloor (r_3 c_1 + (2^l - r_3) c_2)/2^l \rfloor$
   $S = (S^* - \mathbf{a}^\gamma) \bmod N$
```

After Step 2, S^* is the form of $S + a$, where a is a random value. Therefore, the attacker cannot get any information on the real signature S even though he succeeded in skipping the subsequent operations. More detail analysis follows.

Security Analysis
Bellcore-like attack. Suppose that an attacker succeeded in introducing a fault during one of the two exponentiations and also getting rid of the final operation $S = (S^* - \mathbf{a}^\gamma) \bmod N$. Let $\tilde{S}_p^*$ be the faulty exponentiation and $\tilde{S}^*$ be the faulty output of Step 2. Then the attacker will try to compute $\mathrm{GCD}((\tilde{S}^*)^e - m, N)$. However since $(\tilde{S}^*)^e = (\tilde{S} + a)^e$, neither $((\tilde{S}^*)^e - m) \not\equiv 0 \bmod p$ nor $((\tilde{S}^*)^e - m) \not\equiv 0 \bmod q$. Consequently the attacker cannot factorize N.

This is the same when the attacker tries to compute $\mathrm{GCD}(S^* - \tilde{S}^*, N)$. Let a be the random number used in computing S^* and b be the random number used in computing the second faulty signature $\tilde{S}^*$. Then $S^* \equiv S_p + a_p \pmod p$ and $S^* \equiv S_q + a_q \pmod q$. And $\tilde{S}^* \not\equiv \tilde{S}_p + b_p \pmod p$ and $\tilde{S}^* \equiv S_q + b_q \pmod q$. Therefore, since $(S^* - \tilde{S}^*) = (S_p + a_p - \tilde{S}_p - b_p) \not\equiv 0 \bmod p$ and $(S^* - \tilde{S}^*) = (S_q + a_p - S_q - b_q) \not\equiv 0 \bmod q$, the attacker cannot factorize N.

Consideration on skipping operations. Step 3 is consists of two parts. The one is computing c_1, c_2, and γ (Let's say Step 3.1). The other is computing S (We call it as Step 3.2). Our experiments focused on skipping Step 3.2 . Therefore if an attacker succeeds in skipping this, he gets $(S^* + a)$ and receives no valuable information on secret keys. This is the same when he skips both Step 3.1 and Step 3.2. Then how about skipping only Step 3.1? In our modified scheme, γ is initialized with a random number, therefore he gets $(S^* - a^\gamma)$. The only possibility to attack is to make $\gamma = 1$ which is negligible.

Let us consider Giraud's scheme. We first modify SPA-FA-resistant modular exponentiation in order to make the output as the form of $(a + m^{d-1} \bmod N$,

$a + m^d$ mod N), where a is a random value. Similar security analysis is possible, but since Giraud's scheme uses a conditional check-routine, if it is skipped the attack is possible. Therefore avoiding a conditional check-routine is much better. The proposed one is a simple example to avoid our attack (skipping Step 3.2 and skipping both Step 3.1 and Step 3.2). To avoid the attack skipping only Step 3.1 (a conditional check) can be prevented by adding a randomness before the start of exponentiation . Because if an error occurs in a one of exponentiations, then it will affect also a random number used.

Modified SPA-FA-resistant modular exponentiation, SPA-FA-EXP*

```
$a_0 \leftarrow m$
$a_1 \leftarrow m^2 \bmod N$
for $i$ from $n-2$ to 1 do
    $a_{\bar{d}_i} \leftarrow a_{\bar{d}_i} \cdot a_{d_i} \bmod N$
    $a_{d_i} \leftarrow a_{d_i}^2 \bmod N$
$a_1 \leftarrow (\mathbf{a} + a_1 \cdot a_0) \bmod N$
$a_0 \leftarrow (\mathbf{a} + a_0^2) \bmod N$
if (Loop Counter $i$ not modified) & (Exponent $d$ not modified) then
    return $(a_0, a_1)$,
else
    return *error*.
```

Modified Giraud's scheme

```
0.  Choose a random integer $\mathbf{a}$ in $Z_N^*$

1.  $(S_p^*, S_p) \leftarrow$ SPA-FA-EXP$^*(m, d_p, p, \mathbf{a})$
    $(S_q^*, S_q) \leftarrow$ SPA-FA-EXP$^*(m, d_q, q, \mathbf{a})$

2.  $S^* = \mathrm{CRT}(S_p^*, S_q^*)$
    $S = \mathrm{CRT}(S_p, S_q)$
    $S^* = (m \cdot S^* + \mathbf{a}) \bmod (p \cdot q)$
    $S = (S + \mathbf{a} \cdot m) \bmod (p \cdot q)$

3   if $(S^* = S)$ & (Parameters $p$ and $q$ not modified) then
        return $(S - \mathbf{a} - \mathbf{a} \cdot m) \bmod N$
    else
        return *error*
```

5 Conclusions

In this paper, we pointed out the weakness of previous countermeasures against fault attacks on CRT-RSA. Previous countermeasures are all vulnerable since they are constructed without considering this weakness. Furthermore, to the authors' best knowledge, we showed the first (public reported) physical experiment allowing double faults during one execution of the algorithm. Finally, we proposed a simple and almost cost-free method to defeat this attack.

Acknowledgments. The author would like to thank for anonymous reviewers for their valuable comments.

References

1. C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert, Fault attacks on RSA with CRT: Concrete results and practical countermeasures, *Cryptographic Hardware and Embedded Systems – CHES 2002, LNCS V.2523*, pp.260-275, 2002
2. *http://www.atmel.com/product/AVR/*
3. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, The Sorcerers apprentice guide to fault attacks, *Workshop on Fault Diagnosis and Tolerence in Cryptgraphy in association with DSN 2004 – The International Conference on Dependable Systems and Networks*, pp.330-342, 2004
4. D. Boneh, R.A. DeMillo, and R.J. Lipton, On the importance of checking cryptographic protocols for faults, *Advances in Cryptology – EUROCRYPT'97, LNCS V.1233*, pp.37-51, 1997.
5. D. Boneh, R.A. DeMillo, and R.J. Lipton, On the importance of eliminating errors in cryptographic computations, *Journal of Cryptology 14(2)*, pp.101-119, 2001. An earlier version appears in [4].
6. J. Blömer and M. Otto, Wagner's attack on a secure CRT-RSA algorithm recondiered, *Fault Diagnosis and Tolerance in Cryptography – FDTC'06 LNCS V.4236*, pp.13-23, 2006
7. J. Blömer, M. Otto, and J.-P. Seifert, A new CRT-RSA algorithm secure against Bellcore attacks, *10th ACM Conference on Computer and Communications Security*, pp.311-320, 2003
8. M. Ciet and M. Joye, Practical fault countermeasures for Chinese Remaindering based RSA, *Fault Diagnosis and Tolerance in Cryptography – FDTC'05*, pp.124-131, 2005
9. P.-A. Fouque and F. Valette, The doubling attack - why upward is better than downwards, *Cryptographic Hardware and Embedded Systems – CHES'03, LNCS V.2779*, pp.269-280, 2003
10. C. Giraud, Fault resistant RSA implementation, *Fault Diagnosis and Tolerance in Cryptography - FDTC'05*, pp.142-151, 2005
11. C. Giraud, An RSA implementaiton resistant to fault attacks and to simple power analysis, *IEEE Transactions on computers, VOL. 55, NO. 9,* pp.1116-1120, 2006
12. M. Joye, A.K. Lenstra, and J.-J. Quisquater, Chinese remaindering based cryptosystems in the presence of faults, *Journal of Cryptology 12(4)*, pp.241-245,1999.
13. M. Joye, P. Pailler, S.-M. Yen, Secure evaluation of modular functions, *International Workshop on Cryotpology and Network Security 2001*, pp.227-229, 2001

14. M. Joye and S.-M. Yen, The Montgomery powering Ladder, *Cryptographic Hardware and Embedded Systems – CHES 2002, LNCS V.2523*, pp.291-302, 2002
15. P. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, *CRYPTO'96, LNCS V.1109*, pp.104-113, 1996
16. D. Naccache, P.Q. Nguyen, M. Tunstall, and C. Whelan, Experimenting with Faults, Lattices and the DSA, *Public Key Cryptography - PKC 2005, LNCS V.3386*, pp.16-28, 2005
17. A. Shamir, Method and apparatus for protecting public key schemes from timing and fault attacks, United States Patent ♯5,991,415, November 23, 1999. Also presented at the rump session of EUROCRYPT'97.
18. D. Wagner, Cryptanalysis of a provably secure CRT-RSA algorithm, *11th ACM Conference on Computers and Communications Security*, pp.92-97, 2004
19. S.-M. Yen and D. Kim, Cryptanalysis of two protocols for RSA with CRT based on fault infection, *Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'04*, pp.381-385, 2004
20. S.-M. Yen, S. Kim, S. Lim, and S. Moon, RSA speedup with residue number system immune against hardware fault cryptanalysis, *Information Security and Cryptology – ICISC 2001 LNCS V.2288*, pp.397-413, 2001
21. S.-M. Yen, S. Kim, S. Lim, and S. Moon, RSA speedup with Chinese remainder theorem immune against hardware fault cryptanalysis, *IEEE Transactions on Computers 52(4)*, pp.461-472, 2003. An earlier version appears in [20]

CRT RSA Algorithm Protected Against Fault Attacks

Arnaud Boscher[1,⋆], Robert Naciri[2], and Emmanuel Prouff[2]

[1] Spansion,
105 rue Anatole France,
92684 Levallois-Perret Cedex, France
[2] Oberthur Card Systems,
71-73 rue des Hautes Pâtures,
92726 Nanterre Cedex, France
arnaud.boscher@spansion.com, {r.naciri,e.prouff}@oberthurcs.com

Abstract. Embedded devices performing RSA signatures are subject to Fault Attacks, particularly when the Chinese Remainder Theorem is used. In most cases, the modular exponentiation and the Garner recombination algorithms are targeted. To thwart Fault Attacks, we propose a new generic method of computing modular exponentiation and we prove its security in a realistic fault model. By construction, our proposal is also protected against Simple Power Analysis. Based on our new resistant exponentiation algorithm, we present two different ways of computing CRT RSA signatures in a secure way. We show that those methods do not increase execution time and can be easily implemented on low-resource devices.

Keywords: RSA, Chinese Remainder Theorem, Modular Exponentiation, Fault Attacks, Simple Power Analysis, Smart Card.

1 Introduction

In 1997, Boneh, DeMillo and Lipton [1] introduced a new type of cryptanalysis based on error computations: Fault Attacks (FA). Various public-key cryptosystems were concerned but the RSA algorithm was especially targeted. Indeed, Fault Attacks are particularly effective when the Chinese Remainder Theorem (CRT) is applied. Using these techniques, an RSA modulus of arbitrary length can be factorized practically instantly on a PC.

Fault Attacks can be directed at cryptographic embedded devices, like smart cards, as shown in [2]. Straightforward protection mechanisms compute the signature twice, or verify it by performing the inverse operation. Nevertheless, this can be time consuming and further complicated if the corresponding public key is unknown to the device. So, alternative counter-measures, inside the algorithm itself, have been proposed to protect RSA signatures computations against Fault Attacks [3–7]. Unfortunately, many of them have been broken since their publication [2, 8, 9].

⋆ This work was done while the author was with Oberthur Card Systems.

Counteracting FA is not sufficient to ensure the security of an embedded cryptosystem. Indeed, another threat comes from physical leakage during cryptographic computations. A category of attacks, called Side Channel Analysis (SCA), exploits this leakage to retrieve information about sensitive data manipulated by the algorithm. Among these attacks, Simple Power Analysis (SPA) is the easiest to mount in practice and an implementation of a cryptosystem in mobile devices must thwart it. Counteracting FA and SPA attacks at the same time is an issue. Indeed, some counter-measures against SPA can been exploited by elaborate Fault Attacks such as Safe Error Attacks. In fact, it appears that Simple Power Analysis and Fault Attacks (classical and Safe Error) must be simultaneously taken into account when implementing cryptographic algorithms.

The paper is organized as follows. In the next section we briefly recall the RSA cryptosystem, the use of the Chinese Remainder Theorem to speed up generation of RSA signatures and the description of Fault Attacks directed against it. Then, in Sect. 3, we present our method for computing a modular exponentiation protected against Fault Attacks, proving its security in a practical fault model whose relevance to (real life) scenarios is discussed. The new algorithm is used in Sect. 4 to design two CRT RSA implementations resistant to FA and SPA.

2 RSA and Physical Attacks

2.1 RSA Cryptosystem

The public-key cryptosystem RSA [10] involves a public modulus N, which is the product of two large secret primes p and q. The public exponent e is co-prime with $(p-1)\cdot(q-1)$ and the private exponent d is the modular inverse of e modulo $(p-1)\cdot(q-1)$.

An RSA signature S of a message M is computed with the following formula:

$$S = M^d \bmod N.$$

To speed-up the exponentiation on low-resource devices, like smart cards, one usually applies the Chinese Remainder Theorem [11]. The resulting *CRT RSA signature* algorithm is four times faster compared to the classical method. It involves two modular exponentiations and a recombination step using Garner's Algorithm [12]. It needs 5 parameters: the two large primes p and q, the values $d_p = d \bmod p-1$, $d_q = d \bmod q-1$, and the pre-computed value $A = p^{-1} \bmod q$.

Algorithm 2.1. RSA Signature using CRT

```
INPUT: $M, p, q, d_p, d_q, A$
OUTPUT: $S$

1. $S_p \leftarrow M^{d_p} \bmod p$                     //First Exponentiation
2. $S_q \leftarrow M^{d_q} \bmod q$                     //Second Exponentiation
3. $S \leftarrow ((S_q - S_p)\cdot A \bmod q)\cdot p + S_p$   //Garner's Algorithm
4. return($S$)
```

2.2 Simple Power Analysis on RSA Algorithm

By exploiting physical leakage of a device, secret parameters can be retrieved [13, 14] depending on the implementation of the algorithm. Among those Side-Channels Attacks, Simple Power Analysis retrieves information by measuring the power consumption of one execution of the algorithm, whereas Differential Power Analysis (DPA) uses many samples of power consumption and applies statistical techniques to get information. In the following, we particularly focus on SPA attacks. More details about DPA attacks on modular exponentiations, and counter-measures against these attacks, can be found in [15].

To explain how SPA allows one to get information on the secret exponent, let us consider the following basic implementation of a modular exponentiation, known as the Square and Multiply Algorithm, in which the exponent bits are scanned from right to left [16]:

Algorithm 2.2. Right-to-Left Modular Exponentiation

```
INPUT: $M, d = (d_{n-1}, \ldots, d_0)_2, N$
OUTPUT: $M^d \bmod N$

1. $S \leftarrow 1$
2. $A \leftarrow M$
3. for $i$ from 0 to $n-1$ do
4.     if $d_i = 1$ then $S \leftarrow S \cdot A \bmod N$
5.     $A \leftarrow A^2 \bmod N$
6. return($S$)
```

If the executions of a modular square and a modular multiplication have different power consumptions, it has been shown in [13, 14] that information on the value of the secret exponent d can be retrieved. Indeed, if two consecutive modular squares are identified, this means that the exponent bit processed was 0. On the contrary, if a modular multiplication is interleaved between two modular squares, the exponent bit was equal to 1.

To get around this problem, the following algorithm, called Square and Multiply Always, was proposed in [17]:

Algorithm 2.3. SPA Resistant Right-to-Left Modular Exponentiation

```
INPUT: $M, d = (d_{n-1}, \ldots, d_0)_2, N$
OUTPUT: $M^d \bmod N$

1. $S[0] \leftarrow 1$
2. $S[1] \leftarrow 1$
3. $A \leftarrow M$
4. for $i$ from 0 to $n-1$ do
5.     $S[\overline{d_i}] \leftarrow S[\overline{d_i}] \cdot A \bmod N$
6.     $A \leftarrow A^2 \bmod N$
7. return($S[0]$)
```

In Algorithm 2.3, each iteration of the loop involves a modular multiplication whatever the bit-value of the exponent d. Since the sequence of successive operations performed are independent of the key-bits, attacks such as SPA become impossible.

2.3 Fault Attacks on CRT RSA Algorithm

Fault Attacks have been suggested by Boneh *et al.* [1]. They observed that if a device outputs an erroneous CRT RSA signature, an attacker can deduce the private key from this information and the correct signature.

Indeed, let us assume than an error occurs during one of the modular exponentiations of Algorithm 2.1. This results in an incorrect intermediate result, e.g. $\tilde{S}_p$, which will generate an erroneous signature $\tilde{S}$. The faulty signature $\tilde{S}$ and the correct signature S are likely to satisfy $\tilde{S} \not\equiv S \bmod p$ and $\tilde{S} \equiv S \bmod q$. Consequently, if $S - \tilde{S}$ is not divisible by p, the prime number q is revealed by a *gcd* computation : $q = gcd(S - \tilde{S}, N)$.

Remark 1. As noticed in [18], the attack can also be performed without the knowledge of the correct signature: computing $gcd(\tilde{S}^e - M, N)$ will also discover q.

The classical protection against this attack is to verify the computed signature with the public exponent e before sending the signature. The erroneous signature being not returned, the *gcd* computation can no more be computed. However, this can be costly in time (depending on the value of e) and sometimes impossible, if the public key is unknown to the device[1].

In Sections 2.2 and 2.3, we recalled two simple ways of thwarting SPA and FA separately. In the next section, we show that this approach is not enough to obtain a secure implementation of a modular exponentiation.

2.4 Fault Attacks on SPA-Resistant RSA Algorithm

Algorithm 2.3 ensures protection against SPA, but introduces a weakness with respect to another type of Fault Attacks, known as Safe Error, as described in [19].

When an exponent bit equals 0, the result of the *dummy* computation of the modular multiplication is not used any more in the algorithm. Consequently, if a fault is induced on this modular multiplication, an attacker can determine the value of the bit, depending on the correctness or the incorrectness of the modular exponentiation. If the result is correct, the modified modular multiplication was a dummy operation, and so the bit of the exponent was 0. On the contrary, if the result is erroneous, the modified modular multiplication was used in the rest of the algorithm, meaning that the exponent bit was 1.

[1] Computing the public exponent with the knowledge of the 5 CRT parameters is possible but time-consuming on low-resource devices.

This kind of attack can be applied to an RSA signature to recover the private key, irrespective of whether it uses the CRT mode. This kind of cryptanalysis requires more work on the part of the attacker than the analysis discussed in Sect. 2.3 where only one fault was sufficient to obtain the private key. However, it is much more powerful since it thwarts the classical counter-measure consisting in checking the signature before sending it.

The attack described above illustrates the difficulty of thwarting SPA and FA simultaneously. In the following section, we present a new method to compute modular exponentiation resistant against Simple Power Analysis, Fault Attacks and Safe Error Attacks.

3 Exponentiation Resistant to Fault Attacks

3.1 Our Proposal

Our idea consists essentially in modifying an SPA-resistant algorithm by introducing some *coherence test* at the end. This test aims at ensuring that no fault has been induced during the execution of the algorithm. In fact, our reasoning is very close to that proposed in [5] and [20].

Before explaining the core idea of our proposal, let us recall the content of the loop of Algorithm 2.3:

```
4. for $i$ from 0 to $n-1$ do
5.     $S[\overline{d_i}] \leftarrow S[\overline{d_i}] \cdot A \bmod N$
6.     $A \leftarrow A^2 \bmod N$
```

Our idea is based on the three following observations:

– The value A is independent of d. At the end of the algorithm, A satisfies:

$$A = M^{2^n} \bmod N.$$

– The value $S[1]$ is the result of the modular exponentiation of M by the binary complement of d, denoted $\overline{d}$ (and satisfying $\overline{d} = 2^n - d - 1$):

$$S[1] = M^{2^n - d - 1} \bmod N.$$

– Since $S[0]$ equals $M^d \bmod N$, the following relation holds for the content of A after the loop:

$$M \cdot S[0] \cdot S[1] = M \cdot M^d \cdot M^{2^n - d - 1} = M^{2^n} = A \bmod N \ . \tag{1}$$

Equation (1) establishes a relationship between the contents of $S[0]$, $S[1]$, A and M after each loop iteration. So, to ensure that none of the modular multiplications was interfered with (thus counteracting Fault Attacks and Safe Error Attacks), we perform a check between the four values involved in the algorithm. We verify that M, $S[0]$, $S[1]$ and A satisfy Equality (1) before returning $S[0]$:

Algorithm 3.1. SPA/FA Resistant Right-to-Left Modular Exponentiation

INPUT: $M \neq 0, d = (d_{n-1}, \ldots, d_0)_2, N$
OUTPUT: $M^d \bmod N$ or "Error"

```
 1. $S[0] \leftarrow 1$
 2. $S[1] \leftarrow 1$
 3. $A \leftarrow M$
 4. for $i$ from 0 to $n-1$ do
 5.     $S[\overline{d_i}] \leftarrow S[\overline{d_i}] \cdot A \bmod N$
 6.     $A \leftarrow A^2 \bmod N$
 7. if $(M \cdot S[0] \cdot S[1] = A \bmod N)$ and $(A \neq 0)$ then
 8.     return($S[0]$)
 9. else
10.     return("Error")
```

As it can be easily checked, our algorithm is still resistant to SPA: a modular square always follows a modular multiplication, independently of the value of the exponent. We have added two modular multiplications to the original version (Algorithm 2.3). One modular multiplication can be avoided if, at the beginning of the algorithm, $S[1]$ is initialized with the message M. But as we will argue in Sect. 4, the re-use of the message at the end of the algorithm is useful when it comes to protect a CRT RSA that performs exponentiations with Algorithm 3.1. We shall prove in Sect. 3.3 that the coherence check $M \cdot S[0] \cdot S[1] = A \bmod N$ avoids realistic fault attacks when A is not set to zero by the adversary. If A is set to zero, then $S[0]$ and/or $S[1]$ and A shall be null and the coherence check will fail in detecting the fault induction. To prevent such an attack, the verification $A \neq 0$ has been added.

To prove the resistance of our proposal to Fault Attacks, we first have to clarify the capabilities of an attacker. In the following, we define the model in which our algorithm will be proved.

3.2 Attacker Model

As argued in [3] and [8], sensitive applications (e.g. Banking, GSM or Identity Card) cannot make use of countermeasures with ad hoc security but need countermeasures which are provably secure against a precisely modeled adversary. Blömer *et al.* [3], Wagner [8] and, more recently, Lemke-Rust and Paar [21] have introduced adversarial models for Fault Analysis. They consider various natures of faults and attack scenarios with a focus on pervasive computing on low-cost cryptographic devices. The attacker model presented hereafter follows the outlines of those described in [3] and [8]. It is divided into three parts which respectively aim at specifying how the attacker interacts with the device, the kind of variable targeted during the attack and the type of fault.

We shall assume that the attacker is only able to induce one fault per execution of the algorithm (this assumption is discussed in [2]). In [3], Blömer *et al.* identified three different ways to induce faults on an algorithm.

1. Modification of the input parameters [22].
2. Modification of the algorithm execution [23].
3. Modification of the local variables [3].

A powerful adversary is able to induce a fault in the three different manners listed above and nowadays devices are usually provided with hardware mechanisms that render the task of such an adversary as difficult as possible. The adding of redundancy by hardware functions (*e.g.* based on error correcting codes or on hash functions) is often sufficiently effective to protect an implementation against permanent modification of input parameters (first model). Hardware mechanisms can also be successfully involved to guarantee the correctness of an algorithm execution (second model) and they give confidence that the algorithm does not end before all the exponent bits are processed [23]. Even if they are effective and efficient to counteract fault inductions of types 1 and 2, hardware mechanisms are rarely able to thwart attacks based on the perturbation of local variables. Defeating such attacks is usually the main role of software countermeasures. In the rest of the paper, we shall consider an adversary that modifies local variables, assuming that the security against the two other kinds of fault inductions is carried out by the Hardware.

Remark 2. In Appendix A, we propose a slightly modified version of Algorithm 4.1 in which a simple mechanism has been added to counteract some fault injections belonging to the first and the second categories of faults. This version may be used when the effectiveness of some hardware countermeasures is in doubt. It allows to check that the loop has been entirely executed and that the exponent d used during the calculation (and temporarily stored in RAM) has not been modified and equals the exponent d stored in the non-volatile memory of the device.

Let X denote the value of a n-bit local variable and let $\widetilde{X}$ denote the corresponding faulty value. From X and $\tilde{X}$ one can deduce an *error vector* ε such that $\tilde{X} = X + \varepsilon$. The nature of the error vectors ε essentially depends on the adversary type: a strong adversary shall be able to disturb the value of a local variable at a very precise position (*e.g.* a bit modification at a given position), whereas a weak adversary could induce a fault but could not determine its position or its value. Blömer *et al.* exhibited in [3] four different kinds of fault. We recall their classification hereafter.

1. *Precise Bit Errors.* In the strongest scenario, an attacker can change the value of one bit: $\widetilde{X} = X \pm 2^k$ for $0 \leq k \leq n-1$
2. *Precise Byte Errors.* One selected byte is affected by the attack: $\widetilde{X} = X \pm b \cdot 2^k$ for a known $0 \leq k \leq n-8$ and an unknown $0 \leq b \leq 255$
3. *Unknown Byte Errors.* One random byte is affected by the attack: $\widetilde{X} = X \pm b \cdot 2^k$ for a unknown $0 \leq k \leq n-8$ and an unknown $0 \leq b \leq 255$
4. *Random Errors.* An attacker has no knowledge of the modification: $\widetilde{X} = X \pm f(X)$ for $0 \leq f(X) \leq 2^{n-1}$

In our security proof exhibited in the next section, we shall not need to focus on a type of fault in particular and we will prove that our proposal is secure whatever the nature of the fault ε induced by the adversary.

3.3 Security Proof

The message M being assumed to be not null, it can be easily checked that A cannot equal 0 if no fault is introduced. An attack consisting in setting A to zero during the execution of the loop is thwarted by the second test at Step 7. In the rest of this section, we argue that the first test at Step 7 allows to detect any other kind of fault induction in the model described in Sect.3.2.

Wagner proposed in [8] a framework to prove the resistance of an algorithm against Fault Attacks. He suggests that the algorithm be divided into a succession of finite states that correspond to single step computations and to study how faults propagate throughout the algorithm. Such an analysis allows to establish that the fault is either detected by the algorithm or cannot be exploited by the attacker.

The algorithm is split up in such a way that the initial state corresponds to the input of the algorithm and the final state corresponds to the output. All normal transitions between intermediate states are represented by $\rightsquigarrow$. A Fault Attack between intermediate states is symbolized by $\twoheadrightarrow$.

Algorithm 3.1 involves the three variables $S[0]$, $S[1]$ and A. In Wagner's framework, the algorithm execution can be represented by the three following schemes above:

$$1 \rightsquigarrow M^{d_0} \rightsquigarrow M^{d_1 \cdot 2 + d_0} \rightsquigarrow \ldots \rightsquigarrow M^{d}$$
$$1 \rightsquigarrow M^{\overline{d_0}} \rightsquigarrow M^{\overline{d_1} \cdot 2 + \overline{d_0}} \rightsquigarrow \ldots \rightsquigarrow M^{\overline{d}}$$
$$M \rightsquigarrow M^{2} \rightsquigarrow M^{4} \rightsquigarrow \ldots \rightsquigarrow M^{2^n}$$

To prove that the coherence test at the end of our algorithm detects any error during the computation of the three variables, we simulate a fault in a random state $i+1$ for the three schemes above:

1. Attack changing the content of $S[0]$:

$$1 \rightsquigarrow M^{d_0} \rightsquigarrow \ldots \rightsquigarrow M^{\sum_{j=0}^{i-1} d_j \cdot 2^j} \twoheadrightarrow M^{\sum_{j=0}^{i} d_j \cdot 2^j} + \varepsilon \rightsquigarrow \ldots \rightsquigarrow \widetilde{M^d}$$

   The wrong state $M^{\sum_{j=0}^{i} d_j \cdot 2^j} + \varepsilon$ implies a final state $\widetilde{M^d}$ satisfying:

$$\widetilde{M^d} = (M^{\sum_{j=0}^{i} d_j \cdot 2^j} + \varepsilon) \cdot (M^{\sum_{j=i+1}^{n-1} d_j \cdot 2^j}),$$

   that is $\widetilde{M^d} = M^d + \varepsilon \cdot (M^{\sum_{j=i+1}^{n-1} d_j \cdot 2^j})$ which differs from M^d if ε and M are not equal to 0 modulo N.

2. Attack changing the content of $S[1]$. In a similar way, a disturbance of $S[1]$ at any moment results in the following state:

$$\widetilde{M^{\overline{d}}} = M^{\overline{d}} + \varepsilon \cdot (M^{\sum_{j=i+1}^{n-1} \overline{d_j} \cdot 2^j})$$

   which differs from $M^{\overline{d}}$ if ε and M are not equal to 0 modulo N.

3. Attack changing the content of A:

$$M \rightsquigarrow M^2 \rightsquigarrow \ldots \rightsquigarrow M^{2^{i-1}} \twoheadrightarrow M^{2^i} + \varepsilon \rightsquigarrow \ldots \rightsquigarrow \widetilde{M^{2^n}}$$

Contrary to the two previous cases, attacking A at a random state $i+1$ impacts the content of the two others registers $S[0]$ and $S[1]$ at state $i+2$. To better analyze this error propagation, let us rewrite the error in a multiplicative way:

– If M^{2^i} is co-prime with N, we deduce from the additive error ε the multiplicative error β such that:

$$M^{2^i} + \varepsilon = M^{2^i} \cdot (1 + \varepsilon \cdot M^{-2^i}) = M^{2^i} \cdot \beta$$

– If M^{2^i} is not co-prime with N, we denote by z the least common multiple of M and N. The error β is such that:

$$M^{2^i} + \varepsilon = M^{2^i} \cdot (1 + \varepsilon \cdot z \cdot \left(\frac{M}{z}\right)^{-2^i}) = M^{2^i} \cdot \beta$$

So, the different states of the three variables are the following

$$\begin{aligned} 1 \rightsquigarrow M^{d_0} \rightsquigarrow \ldots \rightsquigarrow M^{\sum_{j=0}^{i} d_j \cdot 2^j} \rightsquigarrow M^{\sum_{j=0}^{i+1} d_j \cdot 2^j} \cdot \beta^{d_{i+1}} \rightsquigarrow \ldots \\ 1 \rightsquigarrow M^{\overline{d_0}} \rightsquigarrow \ldots \rightsquigarrow M^{\sum_{j=0}^{i} \overline{d_j} \cdot 2^j} \rightsquigarrow M^{\sum_{j=0}^{i+1} \overline{d_j} \cdot 2^j} \cdot \beta^{\overline{d_{i+1}}} \rightsquigarrow \ldots \\ M \rightsquigarrow M^2 \rightsquigarrow \ldots \rightsquigarrow M^{2^{i-1}} \twoheadrightarrow M^{2^i} \cdot \beta \rightsquigarrow M^{2^{i+1}} \cdot \beta^2 \rightsquigarrow \ldots \end{aligned}$$

and the contents of $S[0]$, $S[1]$ and A finally equal $M^d \cdot \beta^{\sum_{j=i+1}^{n-1} d_j \cdot 2^{j-(i+1)}}$, $M^{\overline{d}} \cdot \beta^{\sum_{j=i+1}^{n-1} \overline{d_j} \cdot 2^{j-(i+1)}}$ and $M^{2^n} \cdot \beta^{2^{n-i}}$ respectively.

When applying our verification formula, we get:

$$\begin{aligned} M \cdot S[0] \cdot S[1] &= M \cdot M^d \cdot \beta^{\sum_{j=i+1}^{n-1} d_j \cdot 2^{j-(i+1)}} \cdot M^{\overline{d}} \cdot \beta^{\sum_{j=i+1}^{n-1} \overline{d_j} \cdot 2^{j-(i+1)}} \\ &= M^{2^n} \cdot \beta^{2^{n-i-1}}, \end{aligned}$$

which is different from the value $M^{2^n} \cdot \beta^{2^{n-i}}$ if the original error ε was not equal to 0.

Remark 3. The error β may have the undesired property that there exists some value k (lower than $n-i-1$) such that $\beta^{2^k} \equiv 1 \mod N$. However, it has been shown in [24] that those values are extremely rare. For instance if N is a RSA modulus equal to the product of two primes p and q, then we have $\beta^{2^k} \equiv 1 \mod N$ iff $2^{n-i-1} \equiv 0 \mod \operatorname{lcm}(f_p, f_q)$, where f_p and f_q are the orders of β modulo p and q respectively. If p and q are such that $p-1$ and $q-1$ are not divisible by large powers of 2, then the probability that this equality holds is comparable to the probability of factoring N by randomly picking one of its prime factors.

Consequently, any error in an intermediate state of the three variables will result in an erroneous result. Thus, we prove that the final check of our algorithm detects any disturbance of any variable during any step of the computation.

4 CRT RSA Resistant to Fault Attacks

In the previous section, we introduced an exponentiation algorithm and proved its security in a realistic fault model. However, even if the two modular exponentiations in the CRT RSA algorithm have not yet been compromised, the correctness of the whole algorithm is not guaranteed. Indeed, it has been shown in [2] that Garner's recombination can be successfully attacked using FA techniques.

The following algorithms use the same principle as the method described in [5] and [20]. A secure modular exponentiation algorithm (Algorithm 3.1) is used to prevent faults during the two exponentiations in the CRT RSA algorithm. Then, additional information given by this secure modular exponentiation is employed to check that the recombination step was not disturbed.

4.1 First Method

Algorithm 3.1 can be used to strengthen the security of a CRT RSA implementation but it has to be slightly modified. Instead of always returning the result of the exponentiation, it returns the three variables if they satisfy Equality (1). Garner's Algorithm is then applied three times, and finally a check is performed to verify that those results satisfy an equality we exhibit below. The goal of this coherence verification is to protect the recombination step.

Proposal. We denote by l the bit-length of the secret moduli. Our CRT-RSA algorithm protected against FA is:

Algorithm 4.1. FA-Resistant RSA Signature using CRT

```
INPUT: $M \neq 0, p, q, d_p, d_q, A$, and $l$ the bit-length of $p$ and $q$
OUTPUT: $S$ or "Error"

1. $(S_p, S'_p, T_p) \leftarrow (M^{d_p} \bmod p, M^{2^l - d_p - 1} \bmod p, M^{2^l} \bmod p)$
2. $(S_q, S'_q, T_q) \leftarrow (M^{d_q} \bmod q, M^{2^l - d_q - 1} \bmod q, M^{2^l} \bmod q)$
3. $S \leftarrow ((S_q - S_p) \cdot A \bmod q) \cdot p + S_p$
4. $S' \leftarrow ((S'_q - S'_p) \cdot A \bmod q) \cdot p + S'_p$
5. $T \leftarrow ((T_q - T_p) \cdot A \bmod q) \cdot p + T_p$
6. if $M \cdot S \cdot S' = T \bmod N$ then
7.     return($S$)
8. else
9.     return("Error")
```

Correctness. We now consider the relevance of the coherence test in Step 6. First, let us denote by $\overline{d}, \overline{d_p}$ and $\overline{d_q}$ the binary complements of the values d, d_p and d_q respectively. They all satisfy:

$$d + \overline{d} = 2^{2l} - 1, \; d_p + \overline{d_p} = 2^l - 1, \; d_q + \overline{d_q} = 2^l - 1.$$

Moreover, by definition of d_p, there exists an integer k such that $d = d_p + k \cdot (p-1)$. Thus, we have:

$$\begin{aligned} d &= d_p + k \cdot (p-1), \\ 2^{2l} - 1 - \overline{d} &= 2^l - 1 - \overline{d_p} + k \cdot (p-1), \\ \overline{d} - 2^{2l} + 2^l &= \overline{d_p} - k \cdot (p-1) \ . \end{aligned}$$

In a same manner, for an integer k' we have:

$$\overline{d} - 2^{2l} + 2^l = \overline{d_q} - k' \cdot (q-1) \ .$$

Due to the Chinese Remainder Theorem, the result of the Garner's recombination of $S'_p = M^{\overline{d_p}} \bmod p$ with $S'_q = M^{\overline{d_q}} \bmod q$ is $S' = M^{\overline{d}-2^{2l}+2^l} \bmod N$. Thus, since $\overline{d}$ equals $2^{2l} - d - 1$, the value S' satisfies $S' = M^{2^l-d-1} \bmod N$.

After multiplying S' by the signature S and the message M (Step 6), we get $M \cdot S \cdot S' = M \cdot M^d \cdot M^{2^l-d-1} \bmod N$ that is $M \cdot S \cdot S' = M^{2^l} \bmod N$.

The fifth step of Algorithm 4.1 computes the value $T = M^{2^l} \bmod N$. Consequently, if no error occurs during the execution of the CRT RSA algorithm, then the four values M, S, S' and T must satisfy the equality:

$$M \cdot S \cdot S' = T \bmod N.$$

Security. The security of Algorithm 4.1 with respect to FA is straightforwardly deduced from the coherence test and the analysis done in Sect. 3 (it thwarts in particular the recent attack [25]). The Square and Multiply Always structure of the algorithm makes it resistant against known-plaintext SPA attacks. SPA-Attacks assuming that the messages can be chosen by the adversary (*e.g.* [26,27]) are out of the scope of this paper. Classical countermeasures such as the randomization of M (see for instance [28]) can be used together with our SPA/FA countermeasure to counteract such attacks by rendering the value of M unpredictable. The use of the message at the end of Algorithm 4.1 (during the last check) protects against modification of the message before one of the two exponentiations and thwarts the attack described in [8]. To insure the validity of the other input parameters of Algorithm 4.1, hardware mechanisms may be used (for instance in order to check the CRC value of each parameter).

Complexity. This method requires adding only two Garner's recombinations and two modular multiplications to the classical CRT RSA algorithm. However, memory consumption is larger. Four l-bit values and two additional $2l$-bit values are required compared to non-protected implementations.

As an alternative, we propose the following algorithm which detects an error with some probability.

4.2 Second Method

Our second proposal uses less memory than the previous one, but the coherence verification is made with a probability error, depending on the bit-length b of a *security parameter* r. This means that an error can remain undetected with a probability equal to $\frac{1}{2^b}$. In the following, we decided to choose a 32-bit parameter b, which is a good compromise between security and efficiency:

Our memory-optimized version of Algorithm 4.1 is:

Algorithm 4.2. FA-Resistant RSA Signature using CRT

INPUT: $M \neq 0, p, q, d_p, d_q, A$, and l the bit-length of p and q
OUTPUT: S or "Error"

```
 1. $(S_p, S'_p, T_p) \leftarrow (M^{d_p} \bmod p, M^{2^l - d_p - 1} \bmod p, M^{2^l} \bmod p)$
 2. $(S_q, S'_q, T_q) \leftarrow (M^{d_q} \bmod q, M^{2^l - d_q - 1} \bmod p, M^{2^l} \bmod q)$
 3. $r \leftarrow$ 32-bit random number
 4. $R_p \leftarrow T_p \bmod r$
 5. $R_q \leftarrow T_q \bmod r$
 6. $S \leftarrow ((S_q - S_p) \cdot A \bmod q) \cdot p + S_p$
 7. if $(M \cdot S \cdot S'_p \bmod p) \neq R_p \bmod r$ then
 8.     return("Error")
 9. if $(M \cdot S \cdot S'_q \bmod q) \neq R_q \bmod r$ then
10.     return("Error")
11. return($S$)
```

The accuracy of the proposed algorithm comes from the definition of the modular exponentiation algorithm employed. The value R_p computed before the recombination is equal to:

$$\begin{aligned} R_p &= T_p \bmod r \\ R_p &= (M^{2^l} \bmod p) \bmod r \ . \end{aligned}$$

The first check computes the value:

$$\begin{aligned} M \cdot S \cdot S'_p &= (M \cdot M^{2^l - 1} \bmod p) \bmod r \\ &= (M^{2^l} \bmod p) \bmod r \\ &= R_p \ . \end{aligned}$$

In a similar way, the last verification step is coherent:

$$\begin{aligned} M \cdot S \cdot S'_q &= (M^{2^l} \bmod q) \bmod r \\ &= R_q \ . \end{aligned}$$

This method requires two additional comparisons with respect to the previous one. But these comparisons are made on values of the same length as p (or q), whereas the comparison in Algorithm 4.1 involves values of same length as N.

Algorithm 4.2 does not require the storage of the l-bit values $M^{2^l} \bmod p$ and $M^{2^l} \bmod q$, during the Garner recombination. Only three 32-bits values must be stored. Instead of T_p and T_q, we store their remainders modulo the 32-bit random number r. Also, the computation and the storage of the public modulus N is no more required.

This optimized version ensures that no attack occurs during the recombination step with a detection probability, which can be parameterized following the bit-size b of the security parameter r.

5 Conclusion

In this paper, we propose a modular exponentiation algorithm that is resistant to Fault Attacks and Simple Power Analysis. We formally prove that this algorithm thwarts classical Fault Attacks and Safe Error Attacks in a realistic and practical fault model. Moreover the timing/memory overhead incurred by the security add-on is quite reasonable. Compared to the classical modular exponentiation algorithms that are resistant to SPA, it requires only two additional modular multiplications.

We show that this exponentiation algorithm can be used to strengthen both versions of the CRT RSA signature algorithm against Fault Attacks. In the first one, only two additional Garner's recombinations and two modular multiplications are needed. And in the second one, only two modular reductions and two modular multiplications are required. These additional overheads do not considerably increase execution time, particularly when compared to the overhead of computing the signature twice over, and is well suited for use on low-resource devices.

Further, the method proposed for computing modular exponentiation in a secure way could be used to compute scalar multiplication of points over the group defined by the points of an elliptic curve.

References

1. Boneh, D., DeMillo, R., Lipton, R.: On the Importance of Checking Cryptographic Protocols for Faults. In Fumy, W., ed.: Advances in Cryptology – EUROCRYPT '97. Volume 1233 of Lecture Notes in Computer Science., Springer (1997) 37–51
2. Aumüller, C., Bier, P., Fischer, W., Hofreiter, P., Seifert, J.P.: Fault attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In Kaliski Jr., B., Koç, Ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems – CHES 2002. Volume 2523 of Lecture Notes in Computer Science., Springer (2002) 260–275
3. Blömer, J., Otto, M., Seifert, J.P.: A New RSA-CRT Algorithm Secure Against Bellcore Attacks. In Jajodia, S., Atluri, V., Jaeger, T., eds.: ACM Conference on Computer and Communications Security – CCS'03, ACM Press (2003) 311–320
4. Ciet, M., Joye, M.: Practical Fault Countermeasures for Chinese Remaindering Based RSA. In Breveglieri, L., Koren, I., eds.: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'05. (2005) 124–132
5. Giraud, C.: Fault Resistant RSA Implementation. In Breveglieri, L., Koren, I., eds.: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'05. (2005) 142–151

6. Shamir, A.: Improved method and apparatus for protecting public key schemes from timing and fault attacks. International Patent Number : WO 98/52319 (1998) Also presented at the rump session of EUROCRYPT'97.
7. Yen, S.M., Kim, S.J., Lim, S.G., Moon, S.J.: RSA Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis. In Kim, K., ed.: Information Security and Cryptology – ICISC 2001. Volume 2288 of Lecture Notes in Computer Science., Springer (2001) 397–413
8. Wagner, D.: Cryptanalysis of a Provable Secure CRT-RSA Algorithm. In Pfitzmann, B., Liu, P., eds.: ACM Conference on Computer and Communications Security – CCS'04, ACM Press (2004) 82–91
9. Blömer, J., Otto, M.: Wagner's Attack on a Secure CRT-RSA Algorithm Reconsidered. In: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'06. (2006)
10. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21** (1978) 120–126
11. Couvreur, C., Quisquater, J.J.: Fast decipherment algorithm for RSA public-key cryptosystem. Electronics Letters **18** (1982) 905–907
12. Garner, H.: The residue number system. IRE Transactions on Electronic Computers **8** (1959) 140–147
13. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Koblitz, N., ed.: Advances in Cryptology – CRYPTO '96. Volume 1109 of Lecture Notes in Computer Science., Springer (1996) 104–113
14. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In Wiener, M., ed.: Advances in Cryptology – CRYPTO '99. Volume 1666 of Lecture Notes in Computer Science., Springer (1999) 388–397
15. Messerges, T., Dabbish, E., Sloan, R.: Power analysis attacks on modular exponentiation in smartcards. In Koç, Ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems – CHES '99. Volume 1717 of Lecture Notes in Computer Science., Springer (1999) 144–157
16. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (1997) Electronic version available at http://www.cacr.math.uwaterloo.ca/hac/.
17. Coron, J.S.: Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems. In Koç, Ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems – CHES '99. Volume 1717 of Lecture Notes in Computer Science., Springer (1999) 292–302
18. Joye, M., Lenstra, A., Quisquater, J.J.: Chinese Remaindering Based Cryptosystems in the Presence of Faults. Journal of Cryptology **12** (1999) 241–246
19. Yen, S.M., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE Transactions on Computers **49** (2000) 967–970
20. Joye, M., Yen, S.M.: The Montgomery Powering Ladder. In Kaliski Jr., B., Koç, Ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems – CHES 2002. Volume 2523 of Lecture Notes in Computer Science., Springer (2002) 291–302
21. Lemke-Rust, K., Paar, C.: An Adversarial Model for Fault Analysis Against Low-Cost Cryptographic Devices. In Breveglieri, L., Koren, I., Naccache, D., Seifert, J.P., eds.: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'06. Volume 4236 of Lecture Notes in Computer Science., Springer (2006) 131–143
22. Yen, S.M., Moon, S., Ha, J.C.: Permanent Fault Attack on RSA with CRT. In Safavi-Naini, R., Seberry, J., eds.: Information Security and Privacy - 8th Australasian Conference – ACISP 2003. Volume 2727 of Lecture Notes in Computer Science., Springer (2003) 285–296

23. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attacks. In Breveglieri, L., Koren, I., eds.: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'04, IEEE Computer Society (2004) 330–342
24. Fumaroli, G., Vigilant, D.: Blinded Fault Resistant Exponentiation. In: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'06. (2006) Available from http://eprint.iacr.org/.
25. Boreale, M.: Attacking Right-to-Left Modular Exponentiation with Timely Random Faults. In Breveglieri, L., Koren, I., Naccache, D., Seifert, J.P., eds.: Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC'06. Volume 4236 of Lecture Notes in Computer Science., Springer (2006) 24–35
26. Fouque, P.A., Valette, F.: The Doubling Attack: Why Upwards is better than Downwards. In Walter, C., Koç, Ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems – CHES 2003. Volume 2779 of Lecture Notes in Computer Science., Springer (2003) 269–280
27. Yen, S.M., Lien, W.C., Moon, S.J., Ha, J.C.: Power Analysis by Exploiting Chosen Message and Internal Collisions – Vulnerability of Checking Mechanism for RSA-Decryption. In Dawson, E., Vaudenay, S., eds.: Progress in Cryptology – Mycrypt 2005. Volume 3715 of Lecture Notes in Computer Science., Springer (2005) 183–195
28. Chaum, D.: Blind signatures for untraceable payments. In Chaum, D., Rivest, R., Sherman, A., eds.: Advances in Cryptology – CRYPTO '82, Plenum Press (1982) 199–204
29. Otto, M.: Fault Attacks and Countermeasures. PhD thesis, Universität Paderborn (2004)

A SPA/FA Resistant Exponentiation with Software Checking of the Exponent

Algorithm A.1. SPA/FA Resistant Right-to-Left Modular Exponentiation - 2nd version

INPUT: $M \neq 0, d = (d_{n-1}, \ldots, d_0)_2, N$
OUTPUT: $M^d \bmod N$ or "Error"

```
 1. $S[0] \leftarrow 1$
 2. $S[1] \leftarrow 1$
 3. $A \leftarrow M$
 4. ectrl $\leftarrow 0$
 5. for $i$ from 0 to $n-1$ do
 6.     $S[\overline{d_i}] \leftarrow S[\overline{d_i}] \cdot A \bmod N$
 7.     ectrl $\leftarrow$ ectrl $+ 2^n \cdot d_i$
 8.     $A \leftarrow A^2 \bmod N$
 9.     ectrl $\leftarrow$ ectrl$/2$
10. if $(M \cdot S[0] \cdot S[1] = A \bmod N)$ and (ectrl $= d$) and $(A \neq 0)$ then
11.     return($S[0]$)
12. else
13.     return("Error")
```

Combinatorial Logic Circuitry as Means to Protect Low Cost Devices Against Side Channel Attacks

Frank Vater, Steffen Peter, and Peter Langendörfer

IHP
Im Technolgiepark 25, 15236 Frankfurt (Oder), Germany
{vater, peter, langend}@ihp-microelectronics.com

Abstract. In this paper we present a clock frequency watch dog that can be realized using a digital standard CMOS library. Such watch dog is required to prevent clock speed manipulations that can support side channel attacks on cryptographic hardware devices. The additional area and power consumed by the watch dog for an AES hardware accelerator are 4,200μm^2 and 2nJ per 128 bit respectively. The physical properties and the use of standard CMOS technology ensure extremely low additional production cost. Thus, our approach is very well suited to improve the security of low cost devices such as wireless sensor nodes.

1 Introduction

Wireless sensor networks (WSN) are becoming more and more popular, and the area of their application is constantly increasing. Their use in military and homeland security applications obviously demands a high level of security. This holds true also for other application areas such as vehicular scenarios [5]. The fact that wireless sensor nodes are exposed to potential attackers requires means to protect them against side channel attacks that exploit physical access to the devices, e.g. against power analysis attacks. These protection mechanisms are normally quite expensive and therefore not used in low cost devices. However, high end smart cards are already equipped with initial protection mechanisms [5].

The main contribution of this paper is the introduction of a circuit that is capable to detect manipulations of the clock frequency, which can be used to simplify differential power analysis attacks. The major benefits of this circuit are extremely small area and energy consumption i.e. 3.5 per cent more energy and approximately 1.0 per cent of the silicon area of the AES (Advanced Encryption Standard [6]) core we used for our experiments. Additionally our approach is a combinatorial logic so that it can be manufactured in a pure CMOS design, which dramatically reduces the costs on integrating this kind of a clock frequency watch dog into crypto hardware. By this, the proposed approach turns formerly unprotected devices to be partly protected devices of level 3 according to the FIPS 140-02 [7].

The rest of this paper is structured as follows. In section 2 we give a overview of the side channel issue and present previous solutions. The AES crypto device that we are evaluating and a successful side channel attack are described in section 3. The a low cost countermeasure is presented and the results are discussed before the paper concludes.

2 Related Work

Side channel attacks as means for revealing secret information of a cryptographic implementation have a long history. In the 1960s intelligence exploited sound and electromagnetic emissions to deduct secret information from cryptographic typewriter [19]. In 1985 van Eck published an analysis of electromagnetic emanations of computer devices [17]. Nowadays timing and power consumption are the major side channels that can be used against cryptographic devices. The first timing attack was published in 1996 [11]. If the processing of a '1' takes a different amount of time than for a '0', key information can be deducted. With a strict separation of data path and control path, as it is possible for modern cryptographic algorithms, that attack does not pose a serious threat anymore. In contrast, power attacks have been a major threat for cryptographic devices. Two kinds of power attacks can be distinguished: the simple power analysis (SPA) and the differential power analysis (DPA). The SPA tries to deduct information directly from the power trace. It can be applied if power consumption for different keys differs a lot. An example for an SPA attack on AES is shown in [12]. The DPA [10] analyses the power consumption of hundreds up to millions of operations and exploits smallest differences of the power trace in order to deduct information. DPA attacks on AES implementations are described in [13][14][4]. Several countermeasures have been presented in [2][3]. A common approach to prevent such analysis is randomization or masking of the performed operations so that small operational differences are covered by intentionally inserted noise. Another approach is the avoidance of any power side channel information. However, solutions for constant power dissipating logic [16, 8] require additional logic gates, additional power consumption and individual design libraries that render these approaches very expensive if not infeasible. In particular for mobile environment, smart cards or in wireless sensor networks where production costs and power consumption must be kept as low as possible such ideas are not practicable.

Additionally most approaches have weaknesses if the circuit is forced into exceptional states. Fault injection, inserted glitches or tampered clock speeds produce errors that finally reveal secret information as described in [13]. Many attacks require a tampered clock frequency in order to force faults. Countermeasures are embedded clock generator or PLLs [15], which are quite expensive and require a lot of additional energy. Thus, we are looking for a low cost mechanism that ensures that the circuit is driven at correct frequency.

3 AES Chip

In this section we first describe the architecture of the evaluated cryptographic accelerator. Afterwards a DPA attack to the AES accelerator is presented.

The IHP Dual^2-Crypto-Chip is a hardware accelerator for the symmetric cipher algorithm AES (Advanced Encryption Standard) and the asymmetric cipher algorithm ECC (Elliptic Curve Cryptography). The chip was manufactured in IHP 0.25μm CMOS technology [9]. Two different interfaces for the connection of the ASIC with a PC were implemented. One is 16-bit PCCARD-Interface, and the other one is a 32-bit PCI-interface. Figure 1 shows the schematic of the interfaces and the AES block. We implemented both interfaces on the demonstration device pursuing the goal to test the device on as many computers and environments as possible. It should be mentioned that the PCI interface gets a system generated clock (33 MHz) while the clock for the PCMCIA interface has to be generated by an external quartz oscillator. That is why we perform our security analysis with the PCMCIA interface and do not consider PCI.

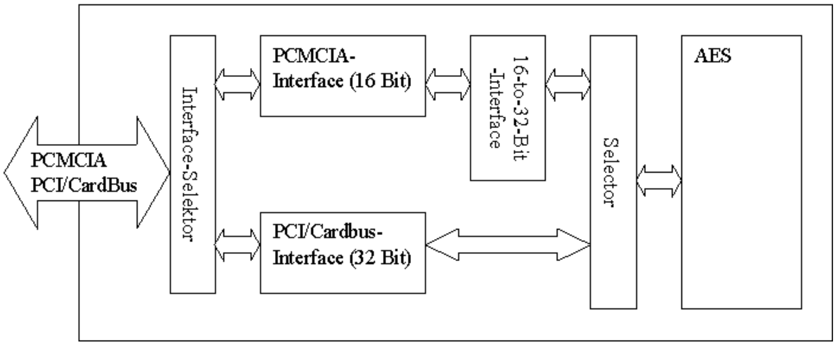


Fig. 1. Block diagram of the IHP Dual^2 crypto chip

3.1 Description of AES Implementation

From side channel security aspects the most interesting block of the IHP Dual^2-Crypto-Chip is the AES block. The AES block investigated in this paper is a similar preceding version of the AES implementation comprehensively described in [18].

The version considered in this paper needs 90 clock cycles to encrypt or decrypt a 128 bit data word using a 128 bit key. As usual the implementation consists of a key generator, an algorithm part and a controller block, which coordinates the key and the algorithm block (see Fig. 2). AES [6] is a block cipher protocol that encrypts each block (i.e. 128 bit) not only in one steps but in several rounds (10). Based on the initial key the AES algorithm needs a new key for every round. The new round key is computed in the key generator block. Once per round the round key is applied in the algorithm block where key and data is

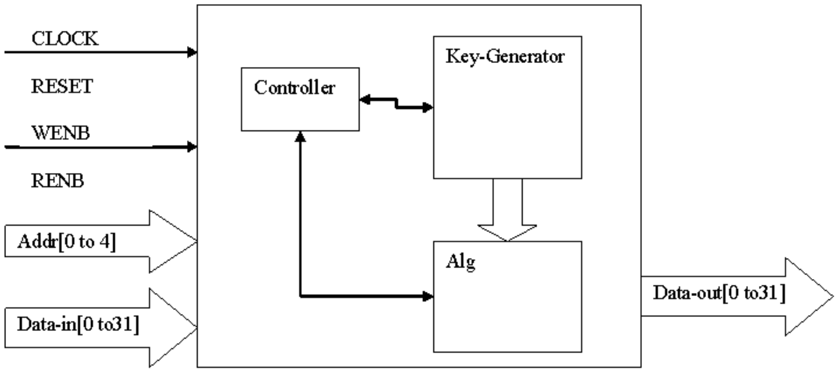


Fig. 2. Block diagram of the AES block

combined to the output stream. In the algorithm block the sub-algorithms add key, shift row, substitution box and mix column are performed once per round. Since the implementation focusses very small low energy devices, the operation are executed successively and not in parallel.

3.2 Successful DPA Attack on the Chip

Considered the AES key is stored on the chip, protocol is known, data streams are fully accessible and the chip can be fully accessed, we want to deduct the internal key exploiting the power consumption as side channel. We also use the fact that we can set the input stream.

In order to deduct the key we observe the initial round of the AES. In other words, we are interested in what the chip is doing when our input data and the stored key are combined for the first time. We are not interested in the actual result of the encryption. In the first AES round the key is bitwise logically XORed with the input data word. The result is stored in a register. The idea is to observe the power that is required for writing the data into the register.

Our approach is to select a constant input value (e.g. all bits ’0’). Then we toggle one bit. The theory is that writing a ’1’ requires more energy than writing a ’0’, because after reset all registers are set to zero [1]. The power consumption can be monitored with an oscilloscope (Agilent 54854A 20GSa/s [1]). Since the power cannot be measured directly, we use a resistor in the power line that generates a voltage drop equivalent to the supply current. Based on our assumption, repeatedly executing the measurement we could filter the noise from other blocks and determine the key. The test environment is shown in Figure 3. It shows a PCMCIA card with the mounted crypto chip, resistor and a port for an external adjustable clock.

[1] That all registers are set to zero after a reset is a property of the circuit that severely weakens the side channel resistance.

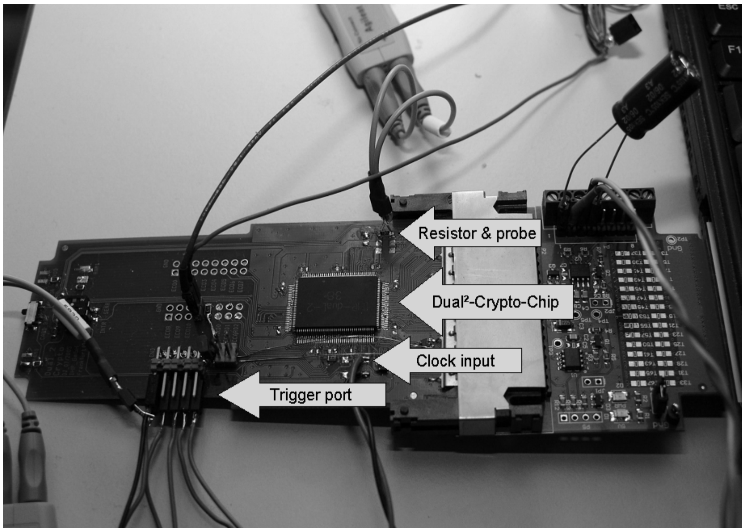


Fig. 3. The tested Dual^2 crypto card with attached measurement equipment

In our experiment in average we could deduct 113 bits out of the 128 bit AES key within 850 iterations of the measuring process. Please consider that even without sophisticated attack statistically the number of correctly guessed bits of the key is already 64.

Our described attack has similarities to the one presented in [14]. However, [14] described an ASIC with separated power supplies for core and I/O-pads, what significantly improves the observability of the core. Our investigated ASIC has only one power supply for both core and pads, what is a more practical scenario for very small and cheap devices as they are applied in mobile environments.

4 Low Cost Countermeasures

Initially we expected that higher capacities of the I/O pads would better cover the action inside the chip. But with the attack assumptions, that are indeed based on the knowledge of the chip design, we could deduct the internal key quite easily. It is a bit surprising and poses the question for efficient countermeasures. In literature several approaches have been described. Many of them require very costly changes in design process and design libraries what is not acceptable in most cases, in particular for cheap small devices.

That is why we are looking for low cost approaches that need as little additional silicon as possible, do not imply much higher power consumption and do not require huge changes in the design chain.

4.1 Clock Stabilization Approach

One thing we could observe in our experiments is that lower clock frequencies improve the observability of the chip significantly. In [14] the clock frequency of the device was reduced to 2 MHz. Our results were also achieved with clock frequencies far under the intended chip specifications (i.e. 10 MHz). We assume that due to capacities on the chip the observability of the internal transitions is reduced with higher clock frequencies - a property that is strengthened by the combined power supply for pads and core.

A straightforward solution for this issue is to ensure that the device is always driven at the intended clock frequency. A standard approach is to integrate a PLL (phase looked loop) into the chip [15]. Though it indeed solves the problem, it is quite expensive. A PLL requires many analogue design elements that are not available in a low cost pure digital design process, so it implies increased costs in developing process and especially for manufacturing.

In the following we propose an approach that can be realized by standard CMOS digital libraries on chip, and is also applicable for FPGAs. The idea is that the gate delay of standard gates is known. With a chain of such gates (e.g. an inverter chain) it is possible to construct a specific delay in the circuit. If the clock edge compared to the output of the inverter chain comes too early or to late, the circuit concludes that the clock frequency has been tampered and causes an invalidation of all results. Since gates in real circuits are not ideal and are depending on temperature and voltage, it is necessary to implement an additional margin. For example if the ideal clock is 50 MHz then one could implement a tolerance of 10 per cent, so that 45 - 55 MHz are still accepted. Figure 4 shows the schematic of such a circuit. The clock is connected to an inverter chain. Two connected inverters are one inseparable buffer. From the inverter chain we fork two signals. The first corresponds to the lower time limit and the second (that is the end of the chain) is the upper threshold. The number of required buffers is computed with following equation:

$$buffers_{short} = \frac{100 - p}{200} \cdot \frac{period}{delay_{per\ buffer}} \tag{1}$$

$$buffers_{long} = \frac{100 + p}{200} \cdot \frac{period}{delay_{per\ buffer}} \tag{2}$$

Where p is the percentage of tolerance.

In our example circuit we want to ensure a clock frequency of 50 MHz with 10% tolerance while the delay per buffer element is 0.0914ns in the considered 0.25μm CMOS technology. Then the required number of buffers needed to supervise the lower boundary of the allowed clock frequency interval is

$$buffers_{short} = \frac{100 - 10}{200} \cdot \frac{20\text{ns}}{0.0914\text{ns}} = 98 \tag{3}$$

and the number of buffers for the upper threshold is 120.

As shown on Figure 4 each of the forked signals is XNORed with the actual clock signal. In the evaluation logic (box at the right) both XNOR results are

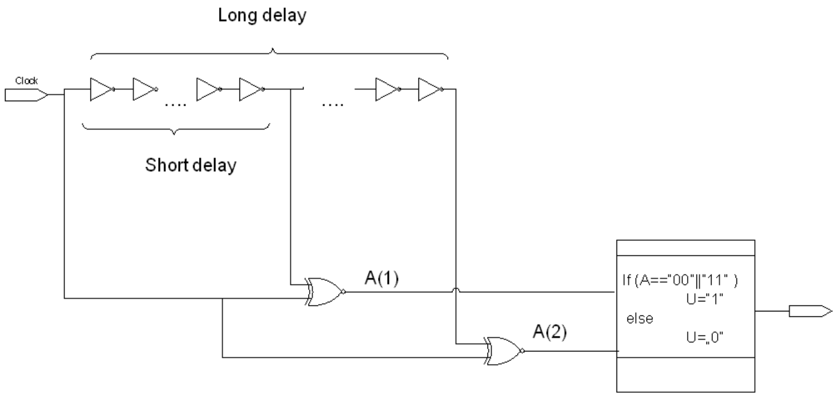


Fig. 4. Structure of the clock watch dog element: Signals A1 and A2 are forked from the delay chain corresponding to the computed threshold time. If A1 equals A2 at toggling clock, it is an indicator for a tampered clock.

compared. We want to see that in the moment the clock toggles, the first forked signal (A1) has been toggled and the second signal not yet. That is, if A1 is '0' and the A2 is '1', or vice versa, then the clock frequency is correct. If in contrast both signals are identical then the clock speed is either too slow or to high because it means that the clock came before or after both threshold signals changed. In case such condition is recognized, the 'untouched' signal is set to '1' what causes an invalidation of the cryptographic circuit. If the clock is assumed as correct the 'untouched' signal is '0'.

4.2 Results

The result of our experiment is shown in Figure 5. If the clock frequency is too slow (on the right) the untouched signal is '1' and consequently the cryptographic operation will not be executed. In the area from 45 to 55 MHz untouched_clk is '0', i.e. it is correct, and higher clock frequencies result in a '1'. However, in the figure it can be seen that higher frequencies sometimes are mistakenly recognized as correct. It happens when the applied clock frequency is a multiple of the correct frequency. That is still an open issue in our approach.

In order to evaluate power and area consumption we integrated our approach into our in house AES implementation. For the IHP 0.25μm CMOS technology the chain of 240 inverters requires 3810μm^2 and the needed additional logic requires 374μm^2. The AES design has an silicon area of 430,000μm^2, thus the additional 4,200μm^2 are reasonable.

The AES encryption of 128 bit with the initial design requires energy consumption of 57 nJ. The same design with secured clock frequency needs 59 nJ, i.e. merely 3.5 percent more energy.

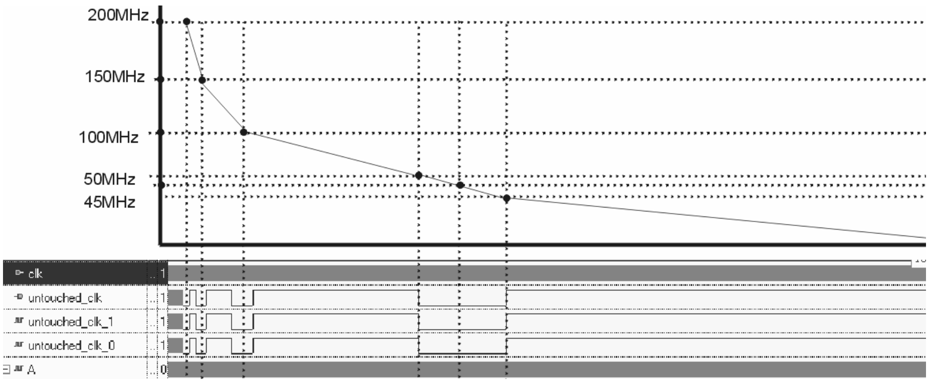


Fig. 5. Waveform of simulated watch dog circuit. On the top is the clock frequency. The row 'untouched_clk' shows the (low active) result of the watch dog. Is it '0' the clock is assumed to ok. Is it '1' it is an indication for a tampered clock.

4.3 Discussion and Further Work

Figure 5 has already shown that high clock frequencies could not be properly detected. Another potential problem is that the production yield will be decreased if the variance of the manufacturing process leads to deviations from the ideal gate delays. It is considerable to implement a delay element that can be calibrated on time after production.

However, beside production deviations, voltage and temperature interfere gate delay times. Lower voltage and higher temperature cause slower circuits. That effect can cause false alerts but can also be exploited to tamper the clock frequency. The PVT (production, voltage, temperature) effects are a serious issue that we have to evaluate in practice after manufacturing silicon circuits with integrated clock watch dog.

Indeed, the proposed mechanism is not a one size fits all solution that prevents from all potential side channel attacks. It is merely a stand alone solution that solves the specific issue of clock frequency manipulation when DPA is used. Thus, it is a piece of a puzzle which can provide complete protection as soon as it is completed. In other words additional means that have to be invented and investigated are still essentially needed. Some potential means are inverse data paths, an increased level of random noise and fixing of design flaws that increase the observability. Also additional capacitors in the pad path could reduce the leaked side channel information in particular if it is guaranteed that the design is driven on sufficiently high clock speeds.

5 Conclusions

In this paper we have presented a clock frequency watch dog realized using combinatorial logic. Our approach causes minimal additional power consumption and negligible area overhead. For our AES design these costs were 3.5 per cent more energy and approximately 1.0 per cent more area. Our simulation results clearly show that the proposed design works very well if a reduction of the clock frequency has to be detected. In the opposite case not all manipulation have been detected. Due to the fact that especially a clock frequency reduction bears a serious risk with respect to DPA, we think that our result is very encouraging.

With the technology presented in this paper we provide a first step towards the realization of partly protected low-cost devices. According to the FIPS 140-02 [7] classification devices using our mechanism can even be grouped into level 3: "devices, that implement tamper evidence and tamper response mechanisms".

To summarize, our approach is a suitable means to significantly improve the security of wireless sensor network for example.

Acknowledgement

This work was partially funded by the German Ministry of Education and Research under grant 01AK060B.

References

1. Agilent Technologies, http://www.home.agilent.com/USeng/nav/-35813.536882578/pd.html. *54854A Infiniium Oscilloscope and InfiniiMax 1132A Probing System*, 2006.
2. Mehdi-Laurent Akkar and Christophe Giraud. An implementation of des and aes, secure against some attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop*, pages 309–318, 2001.
3. J. Blomer, J. Merchan, and V. Krummel. Provably secure masking of aes, 2004.
4. V. Carlier, H. Chabanne, E. Dottax, and H. Pelletier. Electromagnetic side channels of an fpga implementation of aes, 2004.
5. Augusto Julio Domingues Casaca and Dirk Westhoff. Ubisec&sens d0.1 "scenario definition and initial threat analysis". Technical report, June 2006.
6. FIPS. *Advanced Encryption Standard (AES)*. National Institute for Standards and Technology (NIST), November 2001.
7. FIPS. *Security Requirements for Cryptographic Modules*. National Institute for Standards and Technology (NIST), May 2001.
8. J. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor. Security evaluation of asynchronous circuits, 2003.
9. Innovations for High Performance microelectronics, http://www.ihp-ffo.de/24.0.html. *IHP microelectronics: technology*, 2006.
10. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Lecture Notes in Computer Science*, 1666:388–397, 1999.
11. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. *Lecture Notes in Computer Science*, 1109:104–113, 1996.
12. Stefan Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea*, volume 2587 of *Lecture Notes in Computer Science*. Springer, 2003.

13. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, Scotland*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
14. Siddika Berna Örs, Frank Gürkaynak, Elisabeth Oswald, and Bart Preneel. Power-analysis attack on an asic aes implementation. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*. IEEE Computer Society, 2004.
15. S.W. Smith and S.H. Weingart. Building a high-performance, programmable secure processor, tech. report rc 21102. Technical report, IBM T.J. Watson Research Center, 1998.
16. Kris Tiri and Ingrid Verbauwhede. Design method for constant power consumption of differential logic circuits. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 628–633, Washington, DC, USA, 2005. IEEE Computer Society.
17. Wim van Eck. Electromagnetic radiation from video display units: An eavesdropping risk, 1985.
18. Frank Vater and Peter Langendörfer. An area efficient realization of aes for wireless devices. *it - Information Technology*, 3, 2007.
19. Peter Wright. Spycatcher: The candid autobiography of a senior intelligence officer, 1987.

Author Index